HIGH LEVEL SYNTHESIS DESIGN FLOW FOR MULTI PARAMETRIC OPTIMIZATION WITH HYBRID HIERARCHICAL DESIGN SPACE EXPLORATION

by

Zhipeng Zeng

Bachelor of Electrical and Computer Engineering,

Ryerson University, Canada, 2006

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Applied Science

in the Program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2010

Author's Declaration

I hereby declare that I am the sole author of this thesis.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Abstract

Thesis Title:

High Level Synthesis design flow for Multi parametric optimization with Hybrid Hierarchical Design Space Exploration

Submitted by:

Zhipeng Zeng, Master of Applied Science, Electrical and Computer Engineering Program 2010

Directed by:

Reza Sedaghat, Electrical and Computer Engineering Department, Ryerson University

High Level Synthesis (HLS) has definitely bridged the pathway between the Electronic System Level (ESL) and its respective structural block at the Register Transfer Level (RTL). However, the most critical task during HLS is to assess and find a superior architecture from the design space that meets the design objectives. This thesis introduces a novel mechanism for efficient Design Space Exploration (DSE) based on Priority Factor using the Fuzzy search technique to achieve the optimum result. This novel approach is more efficient than traditional DSE approaches and is capable of drastically reducing the number of architectural variants to be assessed for architecture selection. The proposed method, when applied to a number of benchmarks, yielded improved results with remarkable speedup compared to the existing approach. The HLS design flow shown in this thesis uses the proposed approach for DSE with optimization of three parameters, hardware area, execution time and power consumption.

Acknowledgements

I would like to thank very much Dr. Sedaghat and Optimization Problems Research and Applications Laboratory (OPR-AL) members for their continuous contribution and support.

I would also like to express my appreciation to my friends for understanding me and encouraging me to pursue my goal.

I am indebted to my parents and my sister for their constant love and support.

Table of contents

List of Tables

List of Figures

Nomenclature

A	Total Area of the resources
Ri	The resources available for system designing
R_{clk}	The clock oscillator used as a resource providing the necessary clock frequency to the system
N_{Ri}	The number of resource Ri
K_{Ri}	The area occupied per unit resource 'Ri'
WL	Workload to finish all the tasks
L	Latency of execution
T_c	Cycle time of execution
N	Number of data elements to be processed
T_{Ri}	Number of clock cycles needed by resource 'Ri'
T_p	Time period of the clock
p_c	Power consumed per area unit resource at a particular frequency
H(z)	The transfer function of the filter in the frequency domain
$P_{optimal}$	The constraint for Power Consumption
$T_{optimal}$	The constraint for Execution Time

v_i	Number of variants 'i'
P	Total power consumption
T_{exe}	Total execution time
τ_v	Membership value of variant v
τ_B	Membership value of the border variant
$V_{variant}$	Variant's value for a parameter (e.g. area, power and execution time)
Max	The maximum possible value for a certain parameter
Min	The minimum possible value for a certain parameter
$V_{variant}$	Border variant's value for a certain parameter

Chapter 1

Introduction

1.1 Overview on High Level Synthesis

High level synthesis (HLS) is the process of translating the behavioral algorithmic description into a structural building block which realizes the algorithm. During the HLS, the hardware circuit generated from behavior description (as opposed to structure description) consists of a structure composition of data paths, control and memory elements. Therefore, HLS is also known as a transformation from the behavior to structure [1]. HLS can be broadly classified into five major stages: The first stage is the description of the digital system in the form of an algorithm. The second one is the conversion of the abstract behavioral description of algorithm into a sequencing graph or data flow graph. Next is the allocation of the resources for the digital system. The next phase is responsible for allocating many units in the design: the

functional units to execute the operations, storage units to store temporary data, and multiplexers/demultiplexers to switch configuration according to the data flow. The final stage is the interconnection of these structural units using the data transfer information obtained from the sequencing graphs. HLS offers many advantages, such as the productivity gains and the efficient design space exploration. One of the reasons is that performing design space exploration (DSE) at a higher level of abstraction is more profitable than at a lower level of abstraction, or at transistor level. Traditional high level synthesis design methodology is much simpler than modern techniques. The first step of the synthesis involves compiling the behavioral specification into an internal representation. The second step is to apply the high level transformation techniques in order to optimize the behavior as per the desired performance. To realize the structure, the final step is to perform scheduling. Scheduling involves not only the determination of the time at which each operation is executed but also the allocation, which is synthesizing the necessary hardware to perform the operations [26]. Scheduling can have two divisions: time constrained scheduling and resource constrained scheduling. Time constrained scheduling refers to finding the schedule with minimum cost and the same time. The schedule also needs to satisfy the given set of constraints with the given maximum number of control steps. In contrast, resource constrained scheduling refers to finding the fastest possible schedule that satisfies the given set of constraints with the given maximum number of resources. Resource constraints are generally specified by the area occupied by the functional units such as adders/sub-tractors, multipliers, dividers and ALUs. Although the data path of the system

consists of registers and interconnections, they are not considered to be included as resource constraints [19].

For modern multi objective Very Large Scale Integration (VLSI) and System-on-Chip (SOC) designs, analyzing the huge design space has become increasingly significant for translating the algorithmic description of the application into its respective architecture within a reasonable and manageable amount of time. Traditional high level synthesis flow may not suitable for the modern generation of complex VLSI and SOC designs because the conventional design flow takes into account the optimization of two parameters i.e. area and latency. However, the new generation of the system designs requires multi parametric optimization strategies in HLS while simultaneously utilizing rapid and efficient DSE approaches to find the best suitable architecture. This conversion from the algorithm to architecture is known as high level synthesis. This process of translating the algorithmic description into its respective architectural building block involves many interdependent subtasks such as scheduling, allocation and module selection. Therefore, the problem of searching the optimum solution within the huge design space is certainly the conflict between the quality of concurrently optimizing all the selected parameters and the time spent in exploring the huge design space [26].

Furthermore, the complex design space, consisting of numerous functionally equivalent design points, must be analyzed appropriately to meet all the design objectives and the constraints specified as system requirements. For example, portable appliances such as laptops, Personal Digital Assistant (PDA) and MP3 players consisting of embedded applications require high throughput. Embedded applications widely used in the high-end segments i.e. Application

Specific Integrated Circuits (ASICs) for hardware routers and server CPUs require high performance with limited hardware resources. These computation intensive applications should work under certain operational timing constraints at the expense of minimum power consumption. Moreover, such computation intensive applications used in portable devices should also consume minimal power as they operate in power constrained environments. The tradeoffs linked to the choice of architecture selection during DSE must be addressed in order to find a balanced architecture that satisfies all contradictory conditions of the performance requirement [2].

The multi objective design space exploration approaches which are used for tuning the SOC architectures, evaluate sets of different parameters such as latency, area, power, throughput etc. for the target application. These performance requirements vary according to the application requirements. Moreover, recent advancements in the areas of communications and multimedia have led to the growth of a wide array of applications, which require heavy data processing at minimal power expense, at the same time, with limited hardware resources. Such systems require hardware solutions that can satisfy multiple contradictory performance parameters, for example, hardware resources, power consumption and time of execution. An efficient design space exploration strategy is therefore highly critical for the next generation of communication and embedded multimedia platforms.

1.2 Related Work on Design Space Exploration

Many researchers have addressed the problem of efficiently exploring the architecture design space due to time-to-market pressure conditions. According to the work done in [3], which is based on Pareto optimal analysis, the design space is arranged in the form of an Architecture Configuration Graph (ACG) for architecture variant analysis and optimization of performance parameters. Although the results obtained through that method prove quite promising for architectural synthesis of digital systems, the results lacks efficient searching techniques. The problem of obtaining a comprehensive Pareto optimal set is addressed in [4], which suggest the order of efficiency to assist in determining preferences between the different Pareto optimal points. An evolutionary algorithm, namely the Genetic Algorithm (GA) has also been suggested in [5] as a framework for DSE of data paths in high level synthesis. GA is used to concurrently search the space of data path schedules and module/storage selection. An indirect chromosome representation is combined with an efficient heuristic method to derive a solution for chromosome decoding. Moreover, using an evolutionary algorithm in DSE, researchers in [6] successfully evaluate the design for an application specific SOC. Furthermore, authors in [7] and [8] described another approach for DSE in high level systems based on binary encoding of the chromosomes. Again in [9] the GA has been suggested for design space exploration process to yield better results with different encoding technique for creating chromosome.

In addition to the above design space exploration methods, many different optimization techniques exist which provide ESL design space exploration, namely CHARMED [10], MILAN

[2] and Sesame [11]. Several published works in the area of DSE use optimization techniques that consider different objectives, as [12] [13] [14]. Furthermore, authors in [15] propose a heuristic approach based on Pseudo Boolean solvers (PB solvers) and a complete multi-objective PB solver based on their backtracking algorithm. Moreover, in [16], authors present an overview of the Artemis workbench, which provides modeling and simulation strategies for efficient performance evaluation and exploration of heterogeneous embedded multimedia systems. Additionally, authors in [17] developed a model that can assist designers at the system-level DSE stage to explore the utilization of the reconfigurable resources and evaluate the relative impact of certain design choices. Finally, authors in [18] aim to design multi-processor system-on-chip architectures for a given multi-task signal processing application, which would minimize system cost while satisfying the real-time constraints. The approach proposed in this thesis solves the problem of multi objective DSE by introducing an efficient framework based on priority factor method and fuzzy search technique which is unique in itself.

1.3 Summary of contribution

This thesis introduces a novel design space exploration approach during high level synthesis for a multi parametric optimized high level synthesis design flow. The proposed approach finds the accurate architectural variant in a very short exploration time and the multi-parameters are optimized by satisfying the provided operating constraints. The overall contribution can be summarized as follows:

- Developed the mathematic model in order to arrange the design space in increasing order for area
- Developed the mathematic model in order to arrange the design space in increasing order for power consumption
- Developed the mathematic model in order to arrange the design space in decreasing order for execution time
- Developed a searching technique based on Fuzzy Theory
- Created the interconnection of the different block components
- Produced control signals for different circuit components so that they can cooperate with each other
- Created the whole system's circuit diagram and chip layout

The results of the proposed DSE approach when tested on various established HLS benchmarks yielded superior results compared with a current DSE approach for optimization in HLS.

1.4 Organization of the Thesis

After the brief theoretical background on high level synthesis and the introduction of the related work on DSE in high level synthesis, the theoretical background related to the thesis will be introduced in the next chapter. Chapter 2 describes in detail the proposed mathematical

framework for DSE. The proposed fuzzy search technique is presented in Chapter 3. Chapter 4 presents the high level synthesis design flow with the use of the proposed DSE approach while Chapter 5 analyzes the proposed DSE approach and also discusses the implementation of the design flow using the proposed DSE method on FPGA. Finally, the thesis is concluded in Chapter 6.

Chapter 2

Proposed Mathematical Framework for Design Space Exploration

The proposed framework provides the foundation for the design space exploration approach and the HLS design flow. The process of exploring the design space is very tedious and time consuming for the designer. It demands great accuracy and elaborate analysis to determine the optimum design configuration. Therefore, the exploration of the best design variant in an extensive design space within a short period of time is extremely significant. The mathematical framework behind the proposed DSE approach is discussed in the coming section followed by the application of this DSE approach on a real example during HLS.

2.1. Hardware Area Analysis of the Resources

Let the total area of the resources be given as 'A'. R_i denotes the resources available for system; where 1<i<n.

R_{clk} refers to the clock oscillator used as a resource providing the necessary clock frequency to the system. The total area can be represented as the sum of all the resources used for the designed system. Hence total area can be given as:

$$A = \sum A(Ri) \qquad (2.1)$$

The area can be expressed as the sum of the resources i.e. adder/subtractor, multiplier, divider, and also the clock frequency oscillator. Therefore, for a system with 'n' functional resources equation (2.1) can also be represented as:

$$A = (N_{R1} \cdot K_{R1} + N_{R2} \cdot K_{R2} + ... + N_{Rn} \cdot K_{Rn}) + A(R_{clk}) \qquad (2.2)$$

'N_{Ri}' represents the maximum number of the resource '*Ri*' and 'K_{Ri}' represents the area occupied per unit resource '*Ri*' (1<=i<=n).

Applying partial derivative to equation (2.2) with respect to N_{R1}, N_{R2}....N_{Rn} yields equation (2.3), (2.4) and (2.5) respectively as shown below:

$$\frac{\partial A}{\partial N_{R1}} = \frac{\partial((N_{R1} \cdot K_{R1} + ... + N_{Rn} \cdot K_{Rn}) + A(R_{clk})}{\partial N_{R1}} = K_{R1} \qquad (2.3)$$

$$\frac{\partial A}{\partial N_{R2}} = \frac{\partial((N_{R2} \cdot K_{R2} + ...N_{Rn} \cdot K_{Rn}) + A(R_{clk})}{\partial N_{R2}} = K_{R2} \qquad (2.4)$$

... ...

$$\frac{\partial A}{\partial N_{Rn}} = K_{Rn} \qquad (2.5)$$

According to the theory of approximation by differentials [20], the change in the total area can be approximated by the following equation in (2.6):

$$dA = \frac{\partial A}{\partial N_{R1}} \cdot \Delta N_{R1} + \frac{\partial A}{\partial N_{R2}} \cdot \Delta N_{R2} + \ldots + \frac{\partial A}{\partial N_{Rn}} \Delta N_{Rn} + \Delta A(R_{clk}) \qquad (2.6)$$

Substituting the equations (2.3), (2.4) and (2.5) into equation (2.6) yields equation (2.7):

$$dA = \Delta N_{R1} \cdot K_{R1} + \Delta N_{R2} \cdot K_{R2} + \ldots + \Delta N_{Rn} \cdot K_{Rn} + \Delta A(R_{clk}) \qquad (2.7)$$

$$dA = \underbrace{(\Delta N_{R1} \cdot K_{R1})}_{\text{The change of the area contributed by resource R1}} + \underbrace{(\Delta N_{R2} \cdot K_{R2})}_{\text{The change of the area contributed by resource R2}} + \ldots + \underbrace{\Delta N_{Rn} \cdot K_{Rn}}_{\text{The change of the area contributed by resource Rn}} + \underbrace{\Delta A(R_{clk})}_{\text{The change of the area contributed by resource clock}}$$

The above equation indicates the rate of change of the area with respect to the change in the number of the resource R1, R2 ...., R_n. In this analysis the clock oscillator is also considered as a resource which contributes to the area occupied by the hardware resources.

The term priority factor will be used often during the process of exploring the design space in the proposed approach. The priority factor is a determining factor which helps to judge the influence of a particular resource on the variation of the optimization parameters namely area, time of execution and power consumption. This priority factor will be used to organize the architecture design space consisting of the variants in increasing or decreasing order of magnitude. The priority factors for area for the resource R_1, R_2 .....R_n are defined as:

$$PF(R1) = \frac{\Delta N_{R1} \cdot K_{R1}}{N_{R1}} \quad (2.9)$$

$$PF(R2) = \frac{\Delta N_{R2} \cdot K_{R2}}{N_{R2}} \quad (2.10)$$

... ...

$$PF(Rn) = \frac{\Delta N_{Rn} \cdot K_{Rn}}{N_{Rn}} \quad (2.11)$$

$$PF(Rclk) = \frac{\Delta A(Rclk)}{N_{Rclk}} \quad (2.12)$$

The factor defined above determines how the variation in area is affected by the change in the number of certain resources. Hence, the priority factor reflects the rate of change of area with respect to the change in number of resources.

2.2. Analysis for Time of Execution

The term 'workload' of a resource can be defined as the time taken (or clock cycles needed) by each resource to finish its assigned operation during scheduling. Hence the total workload (WL) of all the resources for finishing their respective operations can be represented by (2.13):

$$WL = (N_{R1} \cdot T_{R1} + N_{R2} \cdot T_{R2} + ... + N_{Rn} \cdot T_{Rn}) \quad (2.13)$$

Where NRi represents the number of resource 'Ri' and 'TRi' represents the number of clock cycles needed by the resource 'Ri' (1<=i<=n) to finish each operation.

The execution time is directly proportional to the product of 'the number of times the data elements needs to be processed' and the 'workload' of all the resources. Hence, $T_{exe} \propto (D \cdot WL)$, where 'D' is the number of times the data elements need to be processed for all sets of data. Furthermore, the execution time is also directly proportional to the clock period of the system. Hence we can say that:

$$T_{exe} = K \cdot D \cdot WL \cdot T_p \qquad (2.14)$$

Where, K is the proportionality constant. Substituting (2.13) in (2.14) gives (2.15):

$$T_{exe} = D \cdot K \cdot (N_{R1} \cdot T_{R1} + N_{R2} \cdot T_{R2} + ... + N_{Rn} \cdot T_{Rn}) \cdot T_p \qquad (2.15)$$

From the theory of approximation of differentials the change in execution time is approximated in (2.16).

$$dT_{exe} = \frac{\partial T_{exe}}{\partial N_{R1}} \cdot \Delta N_{R1} + \frac{\partial T_{exe}}{\partial N_{R2}} \cdot \Delta N_{R2} + ... + \frac{\partial T_{exe}}{\partial N_{Rn}} \Delta N_{Rn} + \Delta T_p \frac{\partial T_{exe}}{\partial T_p} \qquad (2.16)$$

Now, applying partial derivative to the (2.15) with respect to NR1.....NRn and Tp will produce the following set of equations:

$$\frac{\partial T_{exe}}{\partial N_{R1}} = \frac{\partial [(N_{R1} \cdot T_{R1} + N_{R2} \cdot T_{R2} + ... + N_{Rn} \cdot T_{Rn}) \cdot T_p \cdot D \cdot K]}{\partial N_{R1}} = T_{R1} \cdot T_p \cdot D \cdot K \qquad (2.17)$$

$$\frac{\partial T_{exe}}{\partial N_{R2}} = \frac{\partial [(N_{R1} \cdot T_{R1} + N_{R2} \cdot T_{R2} + ... + N_{Rn} \cdot T_{Rn}) \cdot T_p \cdot D \cdot K]}{\partial N_{R2}} = T_{R2} \cdot T_p \cdot D \cdot K$$

... ...

$$\frac{\partial T_{exe}}{\partial N_{Rn}} = \frac{\partial[(N_{R1} \cdot T_{R1} + N_{R2} \cdot T_{R2} + ... + N_{Rn} \cdot T_{Rn}) \cdot T_p \cdot D \cdot K]}{\partial N_{Rn}} = T_{Rn} T_p \cdot D \cdot K \qquad (2.18)$$

$$\frac{\partial T_{exe}}{\partial T_p} = \frac{\partial[(N_{R1} \cdot T_{R1} + N_{R2} \cdot T_{R2} + ... + N_{Rn} \cdot T_{Rn}) \cdot T_p \cdot D \cdot K]}{\partial T_p} = (N_{R1} \cdot T_{R1} + N_{R2} \cdot T_{R2} + ... + N_{Rn} \cdot T_{Rn}) \cdot D \cdot K \qquad (2.19)$$

Now, substituting (2.17) (2.18) and (2.19) into (2.16). The substitution yields the following (2.20):

$$dT_{exe} = \Delta N_{R1} \cdot T_{R1} \cdot T_p \cdot D \cdot K + \Delta N_{R2} \cdot T_{R2} \cdot T_p \cdot D \cdot K + .. + \Delta N_{Rn} \cdot T_{Rn} T_p \cdot D \cdot K$$

$$+ D \cdot K \cdot \Delta T_p \cdot (T_{R1} \cdot N_{R1} + T_{R2} \cdot N_{R2} + ... + T_{Rn} \cdot N_{Rn}) \qquad (2.20)$$

Equation (2.20) represents the change in the total execution time with respect to the change in the number of all the resources and the clock period (clock frequency).

$\Delta N_{R1} \cdot T_{R1} \cdot D \cdot K$ = The change of 'Texe' contributed by the change in the number of resource R1. Similarly,

$\Delta N_{Rn} \cdot T_{Rn} \cdot D \cdot K$ = The change of 'Texe' contributed by the change in the number of resource Rn. Finally,

$\Delta T_p \cdot (T_{R1} \cdot N_{R1} + T_{R2} \cdot N_{R2} + ... + T_{Rn} \cdot N_{Rn}) \cdot D \cdot K$ = The change of 'Texe' contributed by the change in clock period (clock frequency).

The priority factor for the 'time of execution' parameter can be defined as it was defined for the area parameter. Therefore, the priority factor for the 'time of execution' parameter in this chapter is defined as follows:

$$PF(R1) = \frac{\Delta N_{R1} \cdot T_{R1}}{N_{R1}} \cdot (T_P)^{\max} \tag{2.21}$$

$$PF(R2) = \frac{\Delta N_{R2} \cdot T_{R2}}{N_{R2}} \cdot (T_P)^{\max}$$

... ...

$$PF(Rn) = \frac{\Delta N_{Rn} \cdot T_{Rn}}{N_{Rn}} \cdot (T_p)^{\max} \tag{2.22}$$

$$PF(Rclk) = \frac{N_{R1} \cdot T_{R1} + N_{R2} \cdot T_{R2} + .. + N_{Rn} \cdot T_{Rn}}{N_{Rclk}} \cdot (\Delta T_p) \tag{2.23}$$

The factors defined above reflect the impact on the average change in the execution time (Texe) with the change in the number of resources. 'D' and 'K' are not included in the expression for the priority factor because they are constants for different resources, and the values of 'D' and 'K' do not affect the order in which the proposed theory organizes the design space for the execution time. The priority factor yields a real number, which suggests the extent to which the change in the number of that particular resource contributes to the change in the execution time. For example, if the priority factor value for R1 is larger than the priority factor value for R2, the execution time will be more sensitive to the change of the number of resource R1 than the change of the number of resource R2.

2.3. Analysis for Power Consumption

For a system with 'n' functional resources the total power consumption (P) of the resources in a system can be represented by the following equation (2.24):

$$P = \sum_{i=1}^{n} (N_{Ri} \cdot K_{Ri}) \cdot p_c \qquad (2.24)$$

$$P = (N_{R1} \cdot K_{R1} + N_{R2} \cdot K_{R2} + .. + N_{Rn} \cdot K_{Rn}).p_c \qquad (2.25)$$

where 'N_{Ri}' represents the number of resource 'Ri' as mentioned before. 'K_{Ri}' represents the area occupied per unit resource 'Ri' (1<=i<=n) and 'p_c' denotes the power consumed per area unit at a particular frequency of operation.

Using the theory of approximation of differentials, the change in the power consumption can be formulated as shown in equation (2.26) :

$$dP = (\frac{\partial P}{\partial N_{R1}} \cdot \Delta N_{R1} + \frac{\partial P}{\partial N_{R2}} \cdot \Delta N_{R2} + ... + \frac{\partial P}{\partial N_{Rn}} \Delta N_{Rn}) + \Delta p_c \cdot \frac{\partial P}{\partial P_c} \qquad (2.26)$$

Now, applying the partial derivative to equation (2.25) will produce the following equations:

$$\frac{\partial P}{\partial N_{R1}} = \frac{\partial[(N_{R1} \cdot K_{R1} + N_{R2} \cdot K_{R2} + ... + N_{Rn} \cdot K_{Rn}) \cdot p_c]}{\partial N_{R1}} = K_{R1} \cdot p_c \qquad (2.27)$$

$$\frac{\partial P}{\partial N_{R2}} = \frac{\partial[(N_{R1} \cdot K_{R1} + N_{R2} \cdot K_{R2} + ... + N_{Rn} \cdot K_{Rn}) \cdot p_c]}{\partial N_{R1}} = K_{R2} \cdot p_c$$

... ...

$$\frac{\partial P}{\partial N_{Rn}} = \frac{\partial[(N_{R1} \cdot K_{R1} + N_{R2} \cdot K_{R2} + ... + N_{Rn} \cdot K_{Rn}) \cdot p_c]}{\partial N_{Rn}} \qquad = K_{Rn} \cdot p_c \qquad (2.28)$$

$$\frac{\partial P}{\partial p_c} = \frac{\partial[(N_{R1} \cdot K_{R1} + N_{R2} \cdot K_{R2} + ... + N_{Rn} \cdot K_{Rn}) \cdot p_c]}{\partial p_c} \qquad (2.29)$$

$$= N_{R1} \cdot K_{R1} + N_{R2} \cdot K_{R2} + ... + N_{Rn} \cdot K_{Rn} \qquad (2.30)$$

Substituting the equations (2.27), (2.28) and (2.30) in equation (2.26) yields the following equation (2.31):

$$dP = (\Delta N_{R1} \cdot K_{R1} \cdot p_c + \Delta N_{R2} \cdot K_{R2} \cdot p_c + ... + \Delta N_{Rn} \cdot K_{Rn} p_c)$$

$$+ \Delta p_c \cdot (K_{R1} \cdot N_{R1} + K_{R2} \cdot N_{R2} + ... + K_{Rn} \cdot N_{Rn}) \qquad (2.31)$$

Equation (2.31) represents the change in the total power consumption with respect to the change in the total number of all the resources and the clock period (clock frequency).

$\Delta N_{R1} \cdot K_{R1} \cdot p_c$ = The change of 'P' contributed by the change in the number of resource R1; Similarly,

$\Delta N_{Rn} \cdot K_{Rn} \cdot p_c$ = The change of 'P' contributed by the change in the number of resource Rn; Finally,

$\Delta p_c \cdot (K_{R1} \cdot N_{R1} + K_{R2} \cdot N_{R2} + ... + K_{Rn} \cdot N_{Rn})$ = The change of 'P' contributed by the change in clock period (clock frequency).

$$PF(R1) = \frac{\Delta N_{R1} \cdot K_{R1}}{N_{R1}} \cdot (p_c)^{\max} \qquad (2.32)$$

$$PF(R2) = \frac{\Delta N_{R2} \cdot K_{R2}}{N_{R2}} \cdot (p_c)^{\max}$$

... .

$$PF(Rn) = \frac{\Delta N_{Rn} \cdot K_{Rn}}{N_{Rn}} \cdot (p_c)^{\max} \qquad (2.33)$$

$$PF(Rclk) = \frac{N_{R1} \cdot T_{R1} + N_{R2} \cdot T_{R2} + .. + N_{Rn} \cdot T_{Rn}}{N_{R_{clk}}} \cdot (\Delta p_c) \qquad (2.34)$$

The priority factors defined above from equations (2.32) to (2.33) indicate the average of change of the total power consumption with respect to the change in the number of resources. For example, equation (2.32) indicates the average of change of the total power consumption of the system when the number of resource R_1 changes (e.g. change in number of adders/subtractors from one to three). The priority factor helps to arrange the architectural variants of the design space in increasing order of magnitude for the parameter of the power consumption. This arrangement further helps the selection of the optimal design point that satisfies all the operating constraints and optimization requirements specified.

The priority factor yields a real number, which suggests the extent to which the change in the number of a particular resource contributes to the change in the total power consumption for the system. The calculated priority factor will be used later in the determination of the arrangement of the design space.

2.4. Organize Design Space with Priority Factor Values

The priority factors are a measurement of the change in different parameters (e.g. area, execution time, power consumption, etc.) with respect to the change in the number of certain resources. The larger the priority factor value for a resource is, the more the resource affects the parameter. Therefore, we can arrange the design space in a form of hierarchic format according to the priority factor values. The resource which affects the parameter the most is assigned to the top of the hierarchic structure, while the resource which affects the parameter the least is located at the bottom of the hierarchic tree. The configuration tree can be build according to the priority factor value of different resources for the parameters such as area, execution time, and power consumption. The resource that has the highest priority factor value will stay on the root of the tree and the resource that has the lowest priority factor value will be the bottom of the hierarchic configuration tree. With that kind of arrangement, the variants will be automatically arranged in certain order (either increase or decrease order).

Chapter 3

Proposed Theory for Fuzzy Searching Technique for DSE

3.1 Theoretical Overview on Fuzzy Logics and Fuzzy Set Theory

The focus of this section is on the proposed theory of fuzzy search technique for design space exploration in high level synthesis. Before deducing the functions of fuzzy search for design space exploration, the general concept behind the proposed theory shall be explored. In the proposed theory, a membership value shall be assigned to each respective element of the set. Compared to the traditional search technique using binary search algorithm, the proposed fuzzy search technique produces the appropriate result within a very short time during DSE. The aim of this chapter is to emphasize the importance that the use of fuzzy search can have in decision

making during DSE and how it can help in drastically reducing the architectural variants to be analyzed for architecture selection.

The fuzzy set theory involves manipulation of the fuzzy linguistic variables [21]. The basic difference between the fuzzy set and classical theory is that in fuzzy set theory every element $x \in U$ is assigned to the value from the interval [0, 1] while in the classical set theory the assignment is from the values of two element set {0, 1}. In fuzzy set theory, the characteristic function is generalized to a membership function that assigns every element 'x' a membership value. The membership function μ_F of a fuzzy set F is a function of the following:

$$\mu_F : U \rightarrow [0,1]$$

3.2 Building the fuzzy set

A graphical representation of the proposed approach takes into consideration that architectural variants in the architectural design space are already organized in increasing or decreasing order. These architectural variants of the design space will be represented in the form of a fuzzy set where each variant will have a certain assigned membership value based on the characterized membership function as shown later. The membership value will be assigned to each variant, while the values of the design space variants are organized in either increasing or decreasing from the left to the right extreme of the fuzzy set. In this theory, only the extreme elements' actual values (which are the minimum and the maximum values or maximum and

minimum values) are calculated at the beginning. The membership value of the variants between the two extremes will be considered to be directly proportional (sorted increasing order or sorted decreasing order) to the position of the variants in the sorted arrangement. Therefore, the membership value of a variant can be calculated by the equations (3.1) or (3.2) for design space arranged in increasing or decreasing orders of magnitude:

$$\tau = \frac{x-\alpha}{\beta-\alpha} \tag{3.1}$$

$$\text{Or, } \tau = \frac{x-\beta}{\alpha-\beta} \tag{3.2}$$

The actual value of the variant is assumed to be proportional to the position of the variant in the sorted arrangement. In equation (3.1) and (3.2), 'x' is the position of the variant; 'τ' presents the membership value of the variant which is the x^{th} element in the sorted arrangement; 'α' and 'β' are the order of the first element and the last element in the same sorted arrangement. Thus, 'α' is equal to 1 and 'β' is equal to the total number of variants in the sorted arrangement. The above function represents a straight line which will aid in finding the border variant, e.g. the first variant which satisfies the execution time constraint and the last variant in the arranged design space which satisfies the specified constraint for area/power.

In all the figures, the x-axis refers to the architectural variants of the design space and the y-axis refer to the actual values and its membership values respectively. 'τ_B' is the membership value of the border variant for the parameter in the architecture space. Similarly, 'τ_V' is the membership value for the variant under test and $V_{Variant}$ is its respective value. Similarly, 'τ_{Min}'

and 'τ_{Max}' are the membership values for the minimum and maximum variants in the architecture space, while 'Min' and 'Max' are their respective values. The increase in trend line for area /power consumption and the decrease in trend line for execution time from left to right extreme of the design space are represented by membership value of each variant. Therefore the actual value of each variant is directly proportional to its associated membership value. An algorithm has been developed to search for the border variant with the given actual value. The graphical representation of this algorithm is shown in Figures 3.1 to 3.4.

3.3 Approach to The Border Variant in The Arranged Design Space

3.3.1 Approach to A Larger Value in An Increasing Trend Line

The trend line shown in Figure 3.1 represents the increase in membership value variants in the design space for the area/power parameters. The membership values in this theory are calculated by applying equation (3.1). After the design space is arranged in increasing order by determining the priority factors, the membership values of each variant are also arranged in increasing order. The actual values of the variants in the design space are directly proportional to the membership values of those variants.

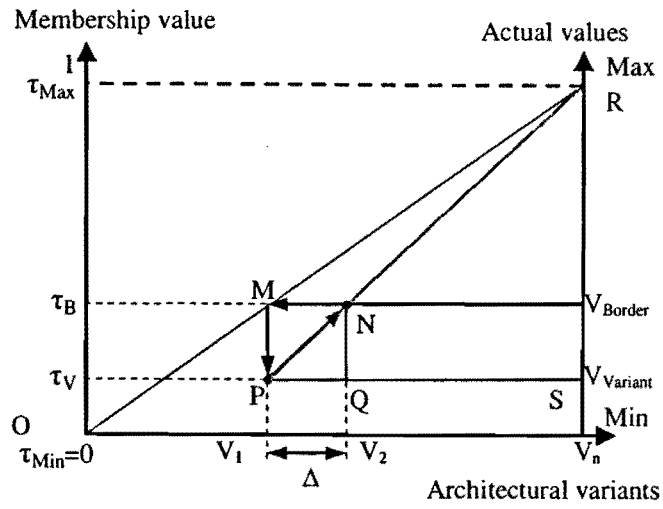

Figure 3.1 Graphical representation of the algorithm for area/power for searching a greater value in the design space

In Figure 3.1, membership value increases for a specific parameter (e.g. actual area increase in the arranged design space). The actual value is approximated by the straight line (OR) drawn from origin (O) to the maximum (R). 'V2' is the border variant with the actual value (e.g. for area/power) V_{Border}. The goal here is to search for V2 with V_{Border} given. 'M' refers to the point in the line (OR) with membership value τ_B which is obtained from the border value (V_{Border}). 'V1' indicates the initial variant obtained from 'M'. 'P' is a point in the straight line corresponding to the actual variant value ($V_{Variant}$) of 'V1'. If the variant value ($V_{Variant}$) is less than the value of border variant (V_{Border}, which could be the value of area/power for V2), then the search should be performed between the points 'P' and 'R'. A second straight line (PR) is drawn to approximate the increasing values for area/power parameter. In this straight line, point 'N' corresponds to the

actual area/power value of variant V2. Therefore, the following function (3.3) can be easily derived.

$$\frac{\tau_{Max} - \tau_V}{\tau_B - \tau_V} = \frac{Max - V_{Variant}}{V_{Border} - V_{Variant}} \qquad (3.3)$$

3.3.2 Approach to A Larger Value in A Decreasing Trend Line

The trend line shown in figure 3.2 represents the decrease in the membership values variants in the design space for execution time parameters. The membership values in this theory are calculated by applying equation (3.2). After the design space is arranged in decrease order by determining the priority factors, the membership values of each variant are also arranged in decrease order. The actual values of the variants in the design space are directly proportional to the membership values of those variants.

In figure 3.2, the membership value decreases for the execution time in the arranged design space. The actual execution time decrease is approximated by the straight line (RU) drawn from the maximum (R) to the minimum (U). 'V2' is the border variant with the value (e.g. for execution time) V_{Border}. 'M' refers to the point in the line (RU) with the membership value τ_B which is obtained from the actual border value (V_{Border}). 'V1' indicates the initial variant obtained from 'M'. 'P' is a point in the straight line corresponding to the actual variant value

($V_{Variant}$) of variant 'V1'. If the variant value ($V_{Variant}$) calculated is less than the value of the boarder variant (V_{Border}, which is the value of execution time of V2), then the search should be performed between the points 'P' and 'R'. A second straight line (PR) is drawn to approximate the decreasing values for the execution time parameter. In this straight line, the point 'N' corresponds to the actual execution time value of variant V2. Therefore, function (3.3) can still be used.

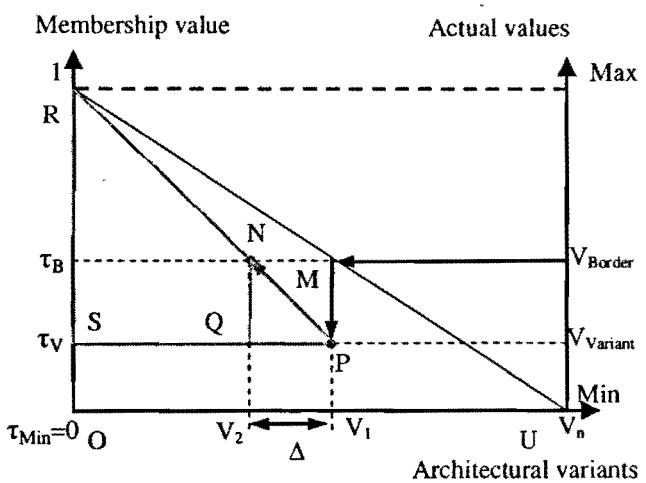

Figure 3.2 Graphical representation of the algorithm for execution time for searching a greater value in the design space

3.3.3 Approach to A Smaller Value in An Increasing Trend Line

The trend line shown in figure 3.3 also represents the increase in the membership values of each variant in the design space for the area/power parameters. The membership values are

calculated according to equation (3.1). After the design space is arranged in increasing order by determining the priority factors, the membership values of each variant are also arranged in increasing order. The actual values of the variants in the design space are directly proportional to the membership values of those variants.

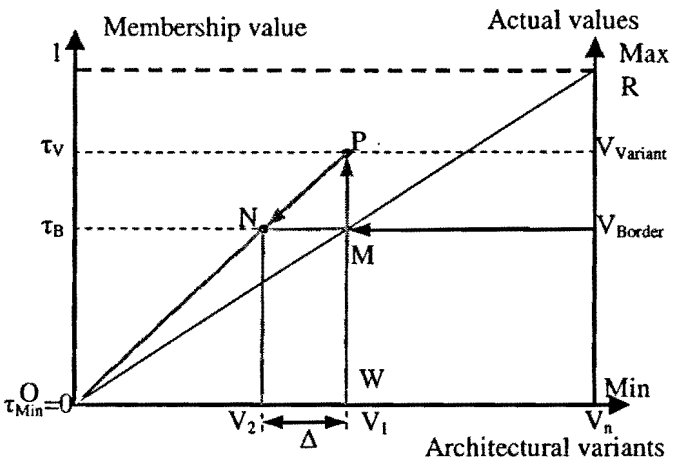

Figure 3.3 Graphical representation of the algorithm for area/power for searching a lesser value in the design space

In figure 3.3, the membership value increases for a specific parameter, i.e. area/power. The actual value is approximated by the straight line (OR) drawn from origin (O) to the maximum (R). 'V2' is the border variant with value (e.g. for area/power) V_{Border}. The goal here is to search for V2 when V_{Border} is given. 'M' refers to the point in the line (OR) with the membership value τ_B which is obtained from the actual border value (V_{Border}). 'V1' indicates the initial variant obtained from 'M'. 'P' is a point in the straight line corresponding to the actual variant value ($V_{Variant}$) of variant 'V1'. If the variant value ($V_{Variant}$) is more than the value of boarder variant (V_{Border}), then the search should be performed between the points 'P' and 'O'. A second straight

line (OP) is drawn to approximate the increasing values for the area/power parameter. In this straight line, the point 'N' corresponds to the actual border value of variant V2. Therefore, the following function (3.4) can be easily derived.

$$\frac{\tau_{Min} - \tau_V}{\tau_B - \tau_V} = \frac{Min - V_{Variant}}{V_{Border} - V_{Variant}} \qquad (3.4)$$

3.3.4 Approach to A Smaller Value in A Decreasing Trend Line

Similar to the previous sections, figure 3.4 represents the decrease trend line for execution time. The actual execution time is approximated by the straight line (RU) drawn from maximum (R) to minimum (U). 'V2' is the border variant with actual value (e.g. for execution time) V_{Border}. 'M' refers to the point in the line (RU) with membership value τ_B which is obtained from the actual border value (V_{Border}). 'V1' indicates the initial variant obtained from 'M'. 'P' is a point in the straight line corresponding to the actual variant value ($V_{Variant}$) of 'V1'. If the variant value ($V_{Variant}$) is more than the value of boarder variant (V_{Border}, which could be the value of execution time for V2), then 'P' should located on the top on 'M', and the search should be performed between points 'P' and point 'U'. A second straight line (PU) is drawn to approximate the decreasing values for execution time parameter. In this straight line point 'N' corresponds to the actual border value of variant V2. Therefore, function (3.3) can also be used.

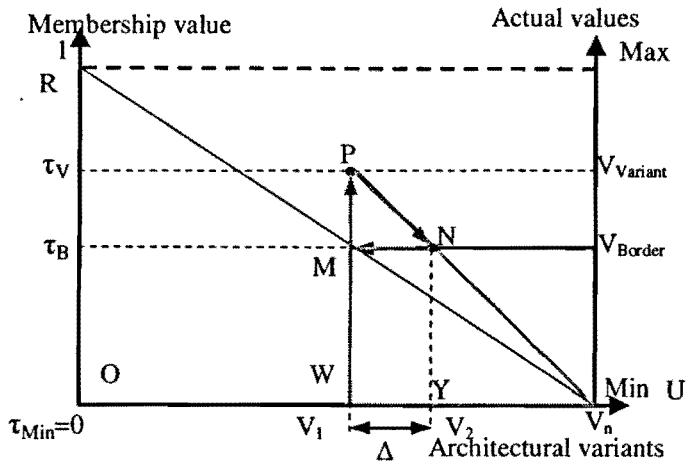

Figure 3. 4 Graphical representation of the algorithm for execution time for searching a lesser value in the design space

3.4 Pseudo-Code for The Proposed Fuzzy Search

In order to find the border variant efficiently, the pseudo-code for this fuzzy technique is described as follows:

Procedure for searching for the border variant (Border)

Step 1 Define the universe of discourse (The fuzzy set)

Step 2 Identify and define the Linguistic variables

Step 3 Assign the approximate membership values (τ) based on the function described in the equation (3.3) or (3.4) for each variant in the universe of discourse based on trendline for that parameter (increasing or decreasing).

Step 4 Calculate the initial membership value (τ_{ini}) based on the function:

$$\tau_{ini} = \frac{V_{Border} - Min}{Max - Min} \qquad (3.5)$$

where τ is the initial membership value corresponding to the border variant (V_{Border}). 'Min' and 'Max' are the minimum and maximum value of the variants for that parameter.

Step 5 Find the variant (V) closest to 'τ_{ini}' in the fuzzy set.

Step 6 Calculate the value of the variant `V`, indicated by $V_{variant}$

Step 7 If $V_{variant} < V_{Border}$ then go to step 8, else go to step 10.

Step 8 Solve the membership value (τ_B) based on the following function:

$$\frac{\tau_{Max} - \tau_V}{\tau_B - \tau_V} = \frac{Max - V_{Variant}}{V_{Border} - V_{Variant}}$$

Step 9 Jump to step 11.

Step 10 Solve the membership value (τ_B) based on the following function:

$$\frac{\tau_{Min} - \tau_V}{\tau_B - \tau_V} = \frac{Min - V_{Variant}}{V_{Border} - V_{Variant}}$$

Step 11 Look for the variant `V` which has the closest membership value to 'τ_B' calculated in step 8 or in step 10.

Step 12 If variant 'V' has already been checked , then

missing a step $V_{variant} == V_{border}$

{If $V_{variant} < V_{Border}$ then look for the unchecked variant with the next higher membership value in the set, and jump to step 13.

Else, if $V_{variant} > V_{Border}$ then look for the unchecked variant with the next lower membership value in the set, and jump to step 13}

Else, variant 'V' has not been checked then go to step 13

Step 13 Calculate the $V_{variant.}$

Step 14 If still the 'Border' is not found then repeat step 7.

Step 15 End

The above procedure successfully determines the border variant for a given performance parameter during searching in DSE. The border variants for area and power consumption indicate that the variant of architecture is the last variant in the design space (design space which is arranged in increasing order of magnitude) to satisfy the V_{Border} specified by the user. The border variant is the first variant in the arranged design space (design space which is arranged in decreasing order of magnitude) that satisfies the V_{Border} specified by the user, for execution time.

Chapter 4

The High Level Synthesis Design Flow Design Flow

The proposed theory behind the framework for DSE will be applied in the upcoming sections. Additionally the fuzzy search method proposed in the previous chapter will be used as a method for searching the best architecture after our design space is organized in increasing or decreasing order based on the priority factor calculation. During the design flow for high level synthesis, three parameters are optimized: hardware area, time of execution, and power consumption. Optimization of multi parameters refers to searching the optimal variant of architecture in the design space that concurrently satisfies the various hard constraints provided, while minimizing the other metric specified in the design problem. Furthermore, variants which satisfy the stringent operating constraints are thereby optimized in terms of area overhead and execution time, as well as quantity of power consumed. The goal of the proposed DSE approach is to find the optimal variant of architecture which satisfies and thereby optimizes, all three

parameters of optimization specified. Figure. 4.1 shows the entire design flow for high level synthesis using the proposed DSE methodology.

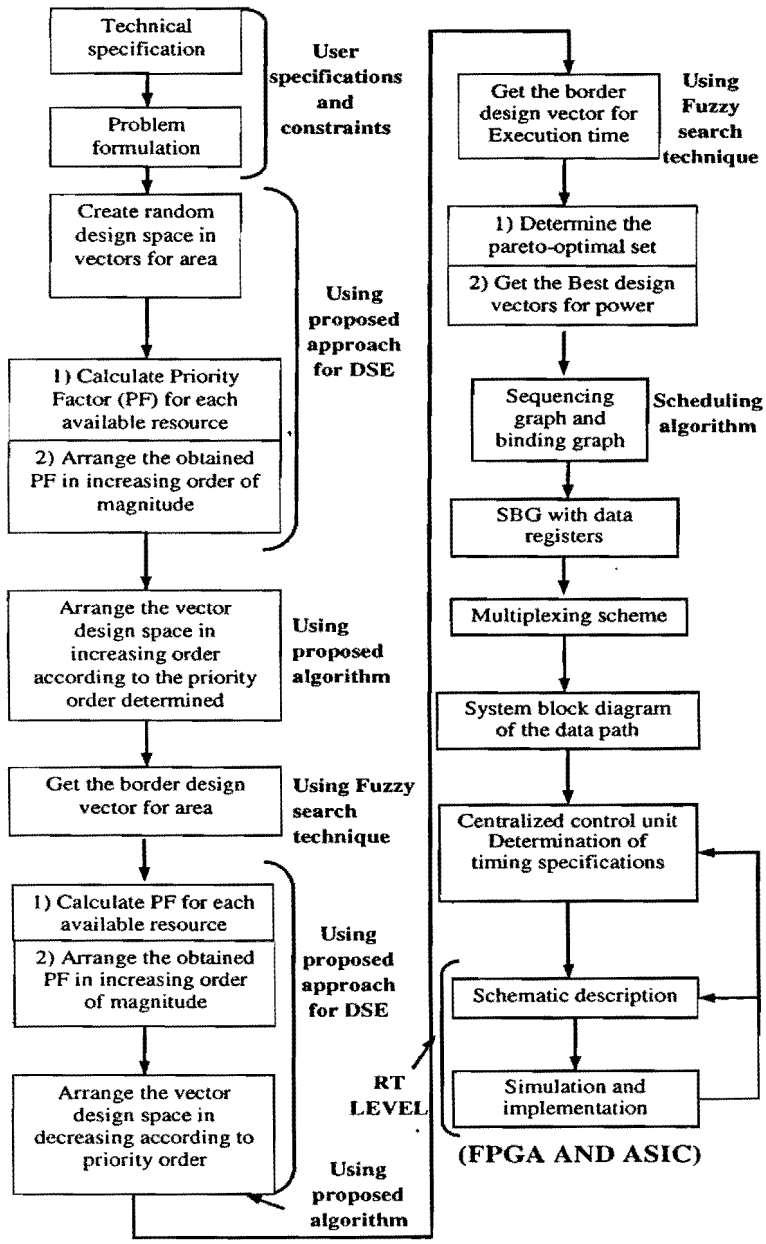

Figure 4. 1 The high level design flow for multi-parametric optimization requirement using the proposed DSE approach

The main idea of the design flow is to first find the optimum configuration architecture of the design, then allocate resources and interconnect different components. Specific details of the design flow will be introduced in this chapter section by section.

4.1. Problem Formulation and Technical Specifications

Table 4. 1 System Specifications

1) Maximum hardware area of resources: 160 area units (a.u)
2) Maximum time of execution: 200μs (For 1000 sets of data)
3) Power consumption: Minimum.
4) Maximum resources available for the system design: a) 3 Adder/subtractor units. b) 3 Multiplier units c) 3 clock frequency oscillators: 24 MHz, 100 MHz and 400 MHz
5) No. of clock cycles needed for multiplier and adder/subtractor to finish each operation: 4 cc and 2cc
6) Area occupied by each adder/subtractor and multiplier: 12 a.u. and 65 a.u. on the chip (e.g. 12 CLB on FPGA for adder/subtractor) 7) Area occupied by the 24MHz, 100MHz and 400 MHz clock oscillators are: 6 a.u., 10 a.u. and 14 a.u respectively.
8) Power consumed at 24MHz, 100MHz and 400 MHz: 10mW/a.u.,32 mW/a.u. and 100mW/a.u. respectively.

This stage marks the beginning of the high level synthesis design flow, beginning with the problem description and the technical specifications provided for the designer. The application should be properly defined with its associated data structure. This phase is critical for the designer and the operational constraints should be clearly defined along with the parameters to be optimized. These specifications act as the input information for the high level synthesis tools. To demonstrate the DSE approach through the design flow, assume the following real specifications for initiating the high level synthesis design flow as shown in Table 4.1.

4.2. Problem Formulation and Description

In the problem formulation stage, the mathematical model of the application is used to define the behavior of the algorithm. The model suggests the input/output relation of the system and the data dependency present in the function. In this chapter the digital IIR Chebyshev filter is used as an example benchmark to demonstrate the DSE method through a high level design flow. The transfer function of a second order digital IIR Chebyshev filter can be given as [22]:

$$H(z)=\frac{Y(z)}{X(z)}=\frac{0.041(1+z^{-1})^2}{1-1.4418z^{-1}+0.6743z^{-2}} \tag{4.1}$$

$$H(z)=\frac{Y(z)}{X(z)}=\frac{0.041+0.082z^{-1}+0.041z^{-2}}{1-1.4418z^{-1}+0.6743z^{-2}} \tag{4.2}$$

$$y(n)=0.041x(n)+0.082x(n-1)+0.041x(n-2)-0.6743y(n-2)+1.4418y(n-1) \tag{4.3}$$

Where x(n), x(n-1) and x(n-2) are the input vector variables for the function. The previous outputs are given by y (n-1) and y(n-2), while the present output of the function is given by y(n). For simplicity, constants 0.041,0.082,0.6743 and 1.4418 are denoted by 'A', 'B', 'D' and 'E' respectively. While x(n), x(n-1), x(n-2), y (n-1) and y (n-2) are denoted by Xn, Xn1, Xn2, Yn1 and Yn2 respectively.

The conversion of the IIR filter function to its digital counterpart is not shown here as it can easily be performed by other well known procedures, namely bilinear transformation and impulse invariant techniques [22].

4.3. Presentation of Variants in the Design Space

Different variants in the architecture design space are represented in the form of vectors consisting of the resources available for the system. For example, Vn = (N_{R1}, N_{R2}, N_{R3}) represents the architecture design space consisting of the resources R1, R2, and R3. If we use R1, R2 and R3 to represent adders/subtractors, multipliers, and clock, then, the variables N_{R1}, N_{R2} and N_{R3} indicate the number of adders/subtractors, multipliers and clock frequencies accordingly. Therefore, according to the specification in Section 4.1, values of N_{R1}, N_{R2} and N_{R3} are as followed: $1<=N_{R1}<=3$, $1<=N_{R2}<=3$, and $1<=N_{R3}<=3$.

The design space which is shown in figure.4.2 represents the different combinations of available resources, which are adder/subtractor, multiplier and choice of clock. The following

section describes the methodology of calculation of the priority factor for area using the equations obtained in section 2.3.

V1 = (1,1,1)	V8 = (1,2,3)	V15= (2,3,2)	V22= (3,1,2)
V2 = (1,2,1)	V9 = (1,3,3)	V16= (2,1,3)	V23= (3,2,2)
V3 = (1,3,1)	V10= (2,1,1)	V17= (2,2,3)	V24= (3,3,2)
V4 = (1,1,2)	V11= (2,2,1)	V18= (2,3,3)	V25= (3,1,3)
V5= (1,2,2)	V12= (2,3,1)	V19= (3,1,1)	V26= (3,2,3)
V6= (1,3,2)	V13 = (2,1,2)	V20= (3,2,1)	V27= (3,3,3)
V7 = (1,1,3)	V14 = (2,2,2)	V21= (3,3,1)	

Figure 4. 2 Design space with all possible resource combinations

4.5. Calculation of the Priority Factor for Available Resources and Arrangement of Design Space in Increasing Order for Area Parameter

The priority factor calculation for area parameter is based on the theory introduced in section 2.1. According to the specification, there are three different types of resources: adder/subtractor, multiplier and clock. Their priority factor values for area can be calculated as following:

For resource adder /subtractor (R1):

$$PF(R1) = \frac{\Delta N_{R1} \cdot K_{R1}}{N_{R1}} = \frac{(3-1)\cdot 12}{3} = 8$$

For resource multiplier (R2):

$$PF(R2) = \frac{\Delta N_{R2} \cdot K_{R2}}{N_{R2}} = \frac{(3-1)\cdot 65}{3} = 43.33$$

For resource clock oscillator (R3):

$$PF(Rclk) = \frac{\Delta A(Rclk)}{N_{Rclk}} = \frac{(14-6)}{3} = 2.67$$

The above factors are a measurement of the change in area with respect to the change in number of certain resources. Here for example, the priority factor values for above different resources indicate that the change in number of multiplier affects the change in the total hardware area the most, while the change in clock frequency from 24 MHz to 400 MHz influences the change in the total hardware area the least. Therefore, we can arrange the design space in a form of hierarchic format, where the resource which affects the parameter the most is located at the top of the hierarchic structure, while the resource which affect the parameter the least is located at the bottom of the hierarchic tree. The configuration tree can be build according to the priority factor value of different resources for area parameter. The resource that has the highest priority factor value will stay on the top of the hierarchic tree and the resource that has the lowest priority factor value will be the bottom of the hierarchic configuration tree. As the area parameter is concerned, the hierarchic tree will have R2 (multiplier) on the top, R1 (adder/subtractor) in the middle, and R3 (clock) on the bottom since the priority factor value for R1 is the highest and the priority factor value for R3 is the lowest. Therefore, the final hierarchic configuration tree is shown in figure 4.3.

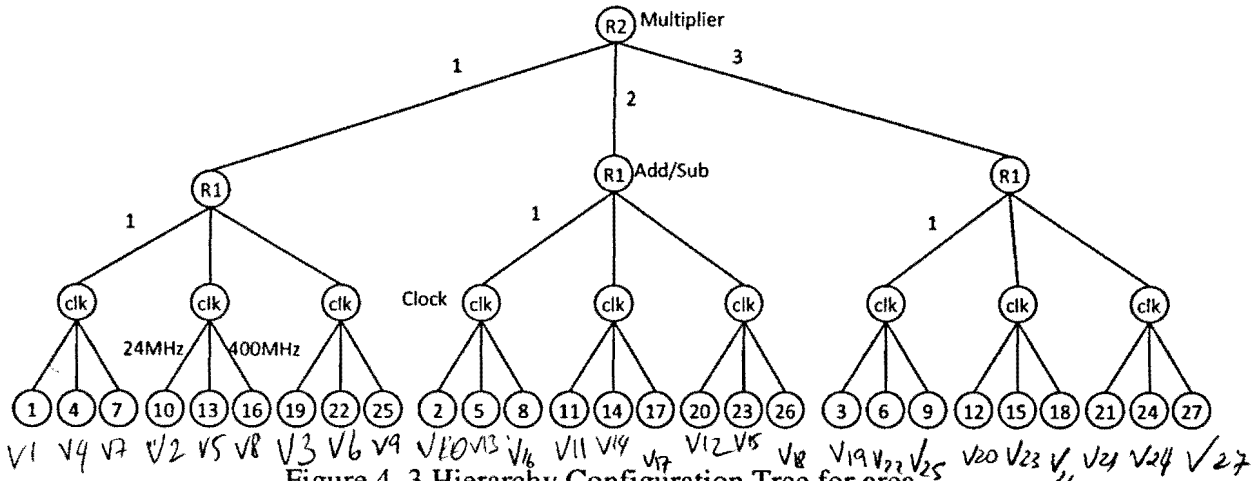

Figure 4. 3 Hierarchy Configuration Tree for area

Based on the above priority factor calculations and hierarchic configuration tree, the variants in design space are organized in increasing orders of magnitude for total hardware area. For example, the arrangement starts from variant 1 on the left to variant 27 on the right, therefore, variant 1 has the lowest area value, and variant 27 has the largest area value. The arrangement of the design variants in increasing order helps to prune the design space for obtaining the border variant for area requirement in the following procedure.

4.6. Fuzzy Search Technique for the Determination of the Border Variant for Area Parameter

The presented fuzzy search technique is applied on the arranged design space. The design space shown in figure 4.3 is arranged in increasing orders of magnitude from the left extreme to the right extreme. This arrangement helps to prune the design space in order to obtain the border variant for the area requirement. The membership values are calculated based on the equation (3.1) since the area value increases from the left extreme variant (V1) to the right extreme variant (V27). For example, the second variant in the hierarchy configuration tree for area is variant V4, therefore, the membership value for area for V4 is:

$$\frac{(2-1)}{(27-1)} = 0.038 \text{, since } x = 2, \alpha = 1, \text{ and } \beta = 27.$$

After the membership value for different variants were calculated, the universe of discourse for area which contains the membership value for area, can be represented by the set shown below.

Large area set (μ_L) =

$$\left\{ \frac{0}{V1}, \frac{0.038}{V4}, \frac{0.077}{V7}, \frac{0.115}{V10}, \frac{0.154}{V13}, \frac{0.192}{V16}, \frac{0.231}{V19}, \frac{0.269}{V22}, \frac{0.308}{V25}, \right.$$
$$\frac{0.346}{V2}, \frac{0.385}{V5}, \frac{0.423}{V8}, \frac{0.462}{V11}, \frac{0.500}{V14}, \frac{0.538}{V17}, \frac{0.577}{V20}, \frac{0.615}{V23}, \frac{0.654}{V26},$$
$$\left. \frac{0.692}{V3}, \frac{0.731}{V6}, \frac{0.769}{V9}, \frac{0.808}{V12}, \frac{0.846}{V15}, \frac{0.885}{V18}, \frac{0.923}{V21}, \frac{0.962}{V24}, \frac{1}{V27} \right\}$$

'Large area' is a linguistic variable for the set defined above, indicating that the membership value increases from 0 to 1 from the first element to the last element.

After the large area set is built, the border variant can be easily found with the help of large area set. According to the specification provided for area, V_{Border} = 160 a.u. while Min = 83 a.u.

and Max= 245 a.u. (according to equation 2.1) are the calculated minimum and maximum values of the variants with the minimum and maximum resources respectively. According to step 4 of the proposed fuzzy search technique, the initial membership value (τ_{ini}) calculated based on the values of Min and Max for area according to equation (3.5), is found as the following:

$$\tau_{ini} = \frac{160-83}{245-83} = 0.475$$

From the set μ_L, it can be seen that the variant which has the membership value closest to 0.475 is the variant V11 whose membership value is 0.462. Therefore, the area value for V11 will be calculated in order to compare it with the border value. The area of V11 turns out to be 160 a.u. which is not bigger than the border value. Hence, the search should continue in order to find a variant with larger area value compared to V11. According to the pseudo-code in section 3.4, the equation (3.3) is applied and the next member ship value is calculated as 0.462. Therefore, V14 is selected as the next variant whose area value will be compared with the border value. The area value of V14 is calculated as 164 a.u. which is larger than the border value. Therefore, the border variant is found to be V11 because V11 satisfies the area required, while the variant V14 which is just next to V11 in the hierarchy configuration tree does not satisfy. It means that in the hierarchy configuration tree, the variants V1 to V11 satisfy area requirement and the rest should be ignored because they do not satisfy the area requirement. The detailed process of fuzzy search is shown table 4.2.

there is no guarantee that V1 to V11 doesn't contain variant that

Table 4. 2 The variants obtained for area when fuzzy search technique was applied

Equations for obtaining the calculated membership values	Calculated membership values(τ)	Variants corresponding in the set according to the calculated 'τ'	Area	Decision based on the V_{Border}
$\tau_{ini} = \frac{140-83}{245-83}$	$\tau_{ini} = 0.475$	0.462/V11	$A^{11} = 2*12 + 2*65 + 6$ = 160 a.u.	$A^{11} <= V_{Border}$, search down in the design space
$\frac{1-0.462}{\tau_B - 0.462} = \frac{245-83}{160-83}$	$\tau_B = 0.462$	0.500/V14	$A^{14} = 2*12 + 2*65 + 10$ = 164 a.u.	Stop

As shown in the table, the fuzzy search technique finds the border variant in just two iterations. The border variant obtained is variant 11. This value indicates the last variant in the space which satisfies the constraint for area requirement (V_{Border}).

4.7. Calculation of the Priority Factor for Each Available Resource for Execution Time Parameter

After the border variant for area is found, similar procedure for the border variant for execution time can be carried. We can arrange the design space in a form of hierarchy configuration tree for execution time. The resource which affects the execution time the most is located at the top of the hierarchic structure, while the resource which affects the execution time the least is located at the bottom of the hierarchic tree. In order to create the hierarchy configuration tree for execution time, the priority factor value for different resources for execution time should be calculated first.

For resource adder/subtractor (R1):

$$PF(R1) = \frac{\Delta N_{R1} \cdot T_{R1}}{N_{R1}} \cdot (T_p)^{\max} = \frac{(3-1)\cdot 2}{3} \cdot (0.0416) = 0.055$$

For resource multiplier (R2):

$$PF(R2) = \frac{\Delta N_{R2} \cdot T_{R2}}{N_{R2}} \cdot (T_p)^{\max} = \frac{(3-1)\cdot 4}{3} \cdot (0.0416) = 0.1109$$

For resource clock oscillator (Rclk):

$$PF(R_{clk}) = \frac{N_{R1} \cdot T_{R1} + N_{R2} \cdot T_{R2}}{N_{Rclk}} \cdot (\Delta T_p) = \frac{(3\cdot 2 + 3\cdot 4)\cdot (0.0416 - 0.0025)}{3} = 0.2346$$

The factors determined above indicate the extent of the change in execution time with the change in the number of a specific resource. For instance, since the priority factor value for adder/subtractor is the smallest and the priority factor value for clock is the largest, according to the above analysis the change in the number of adder/subtractor affect the change in execution time the least, while the change in clock frequency from 24 MHz to 400 MHz affect the change in execution time the most. Similarly, the change in execution time cost by the change in the number of multiplier is lesser than that cost by the change in clock frequency. On the calculation of priority factor, the minimum clock frequency is used because at this frequency the clock period is the maximum. Hence the change in the number of a specific resource at the maximum clock period influences the change in the cycle time the most, when compared to the change in cycle time at other clock periods.

After the calculation of priority factor for different resources, the hierarchy configuration tree can be build according to the priority factor value of different resources for execution time parameter. Similar to the configuration tree for area, the resource that has the highest priority factor value for execution time will stay on the top of the hierarchic tree and the resource that has the lowest priority factor value will be the bottom of the hierarchic configuration tree. As the execution time parameter is concerned, the hierarchic tree will have R3 (clock) on the top, R2 (multiplier) in the middle, and R1 (adder/subtractor) on the bottom since the priority factor value for R3 is the highest and the priority factor value for R1 is the lowest (PF(R3, clock) > PF(R2, multiplier) >PF(R1, add/sub)). The final hierarchic configuration tree for execution time is shown in Figure 4.4.

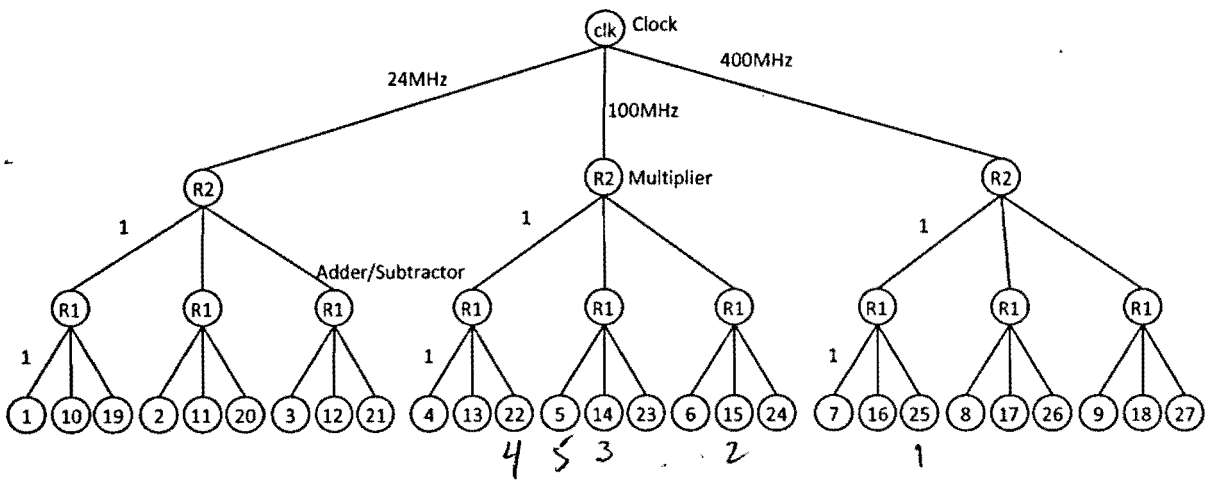

Figure 4. 4 Hierarchic Configuration Tree for execution time

Based on the above priority factor calculations and hierarchic configuration tree for execution time parameter, the variants in design space are arranged in decreasing orders of magnitude for execution time. For example, the arrangement starts from variant 1 on the left to variant 27 on the right, therefore, variant 1 has the largest value for execution time parameter, and variant 27 has the lowest value for the same parameter. This arrangement of the design variants in decreasing order helps to prune the design space for obtaining the border variant for execution time parameter requirement in the design space exploration process.

4.8. Determination of the Border Variant for the Execution Time Parameter

Similar to the search for the border variant for the area parameter, the search for the execution time parameter will be performed in the arranged design space for execution time. As explained in the last section, the design space is arranged in decreasing orders of magnitude for execution time as shown in figure 4.4. Therefore, the membership values of the variants for execution time can be calculated based on the equation (3.2). For example, the second variant in the hierarchy configuration tree for execution time is V10, therefore, the membership value for execution for V10 is: $\frac{(2-27)}{(1-27)} = 0.962$, since x =2, $\alpha = 1$, and $\beta = 27$.

After the membership value for different variant were calculated, the universe of discourse for execution time parameter which contains the membership value for execution time can be represented as:

Small time of execution set (μ_S) =

$$\{ \frac{1}{V1}, \frac{0.962}{V10}, \frac{0.923}{V19}, \frac{0.885}{V2}, \frac{0.846}{V11}, \frac{0.808}{V20}, \frac{0.769}{V3}, \frac{0.731}{V12}, \frac{0.692}{V21}, \frac{0.654}{V4}, \frac{0.615}{V13}, \frac{0.577}{V22}, \frac{0.538}{V5}, \frac{0.500}{V14}, \frac{0.462}{V23}, \frac{0.423}{V6}, \frac{0.385}{V15}, \frac{0.346}{V24}, \frac{0.308}{V7}, \frac{0.269}{V16}, \frac{0.231}{V25}, \frac{0.192}{V8}, \frac{0.154}{V17}, \frac{0.115}{V26}, \frac{0.077}{V9}, \frac{0.038}{V18}, \frac{0}{V27} \}$$

The 'Small time of execution' is a linguistic variable for the set defined above, indicating that the membership value decreases from 1 to 0 from the first element to the last element.

The fuzzy search technique, which is also used for finding the border variant for area parameter, will again be applied in this section to find the border variant for execution time parameter. The initial membership value (τ_{ini}) is calculated based on the Min and the Max values of execution time according to the equation (3.5). According to the specification, V_{Border} = 200 µs, while Min = 20.01µs, Max= 833.41 µs are the calculated minimum and maximum values (according to equation 13) of the variants with maximum and minimum resources respectively. The detail process of finding the border variant (V5) is shown in table 4.3.

This value indicates the first variant in the design space, which satisfies the constraint for the execution time specified (V_{Border}).In the arranged design space, only the variants to the left of the border variant (V5) satisfy the execution time requirement while others do not (to the left of V5 in the arranged design space).

Table 4. 3 The variants obtained for execution time when fuzzy search technique was applied

Equations for obtaining the calculated membership values	Calculated membership values(τ)	Variants corresponding in the set according to the calculated 'τ'	Execution time	Decision based on the V_{Border}
$\tau_{ini} = \frac{200-20.01}{833.41-20.01}$	$\tau_{ini} = 0.2213$	0.231/V25	T_{exe}^{25} = (22 +(1000-1)*20) *0.0025 = 50.005 μs	$T_{exe}^{25} < V_{Border}$, search up in the design space
$\frac{1-0.231}{\tau_B-0.231} = \frac{833.41-50.005}{200-50.005}$	$\tau_B = 0.378$	0.385/V15	T_{exe}^{15} = (12 +(1000-1)*8) *0.01 = 80.04 μs	$T_{exe}^{15} < V_{Border}$, search up in the design space
$\frac{1-0.385}{\tau_B-0.385} = \frac{833.41-80.04}{200-80.04}$	$\tau_B = 0.483$	0.500/V14	T_{exe}^{14} = (14 +(1000-1)*10) *0.01 = 100.04μs	$T_{exe}^{14} < V_{Border}$, search up in the design space
$\frac{1-0.500}{\tau_B-0.500} = \frac{833.41-100.04}{200-100.04}$	$\tau_B = 0.568$	0.577/V22	T_{exe}^{22} = (22 +(1000-1)*20) *0.01 = 200.02 μs	$T_{exe}^{22} > V_{Border}$, search down in the design space
$\frac{0-0.577}{\tau_B-0.577} = \frac{20.01-200.02}{200-200.02}$	$\tau_B = 0.577$	0.538/V5 (Since V22 has been checked so check V5)	T_{exe}^{5} = (14 +(1000-1)*10) *0.01 = 100.04 μs	Stop

4.9. Determination of the Pareto-Optimal Set of Design Architecture

After obtaining the border variants for both the area and execution time parameters, the next step is to find the design variants which simultaneously satisfy the specifications for both area and execution time needed for the design.

The border variant for area parameter is V11. Therefore, all the variants from V1 to V11 in figure 4.3 should be considered for future analysis. They can be represented by a set A = {V1, V4, V7, V10, V13, V16, V19, V22, V25, V2, V5, V8, V11}. Similarly, as the execution time parameter is concerned, the border variant for execution time parameter is V5. Therefore, all the variants from V5 to V27 in figure 4.4 should be considered for future analysis and they can be represented by a set B = {V5, V14, V23, V6, V15, V24, V7, V16, V25, V8, V17, V26, V9, V18, V27}. The variants which optimize both parameters (area and execution time) by satisfying their constraints are variants A∩B= {V5, V7, V16, V25, V8}. Hence, the Pareto optimal set C = {V5, V7, V16, V25, V8}.

In order to optimize the third parameter, the power consumption parameter, the hierarchy configuration tree for power consumption can also be built based on the priority factor values for power consumption for different resources (adder/subtractor, multiplier, and clock). Similar to the procedure described in section 4.5, building the hierarchy configuration tree for power consumption parameter can be very easily obtained by using the equations (2.32) to (2.34). Therefore, the hierarchy configuration tree for power consumption is shown in figure 4.5.

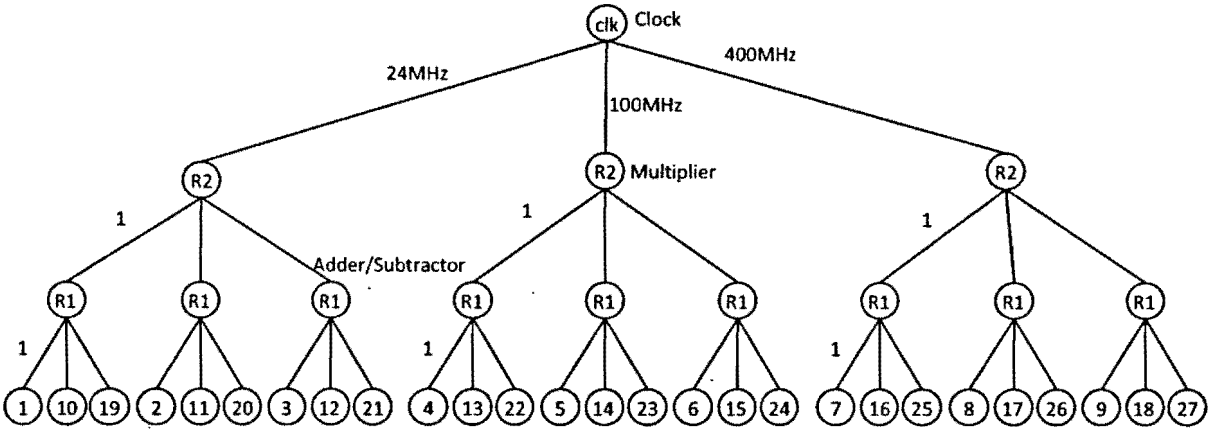

Figure 4. 5 Hierarchic Configuration Tree for power consumption

With the help of hierarchy configuration tree, the variants for power consumption are then arranged in an increasing order of magnitude using the proposed method. According to the specification (see section 4.1) provided, the variant with the minimum power consumption should be selected. When the variants are arranged in an increasing orders of magnitude, the first variant in the order is guaranteed to be the variant with the minimum value (i.e. minimum power consumption) with the value increasing respectively with each next variant of the set. Among all the elements in the Pareto set C, V5 is the first element in the hierarchy configuration tree for power consumption parameter. Therefore, this variant is regarded as the best variant for the system designing as it concurrently satisfies the specification of all the three optimization parameters.

4.10. The Scheduling of Operations through the Sequencing Graph for the Best Variant Obtained

Scheduling is mainly classified into two categories: resources constraint and time constraint scheduling. For resources constraint scheduling, operations will be scheduled within the limited resources, while other parameters (e.g. latency) will be optimized. On the other hand, time constraint scheduling is a process that assigns the time slot for every operation while fixing the timing length (latency) in such a way so that the synthesized hardware structure meets the timing requirement.

The scheduling for the best variant is based on resources constraint scheduling, as the number of resources are fixed for a specific variant. Many scheduling algorithms exist for scheduling of operations, such as As Late as Possible (ALAP), List scheduling, Force Directed scheduling, As Soon As Possible (ASAP) algorithm [23] [24], etc. but the ASAP algorithm was selected as scheduling algorithm so that the operations can be done as soon as the resources become free.

The data flow graph is a directed graph with the edges showing the data flow among operations. The designed function is represented in the form of data flow graph (shown in figure 4.6) before the scheduling procedure starts, so that the dependence among different operations is evident.

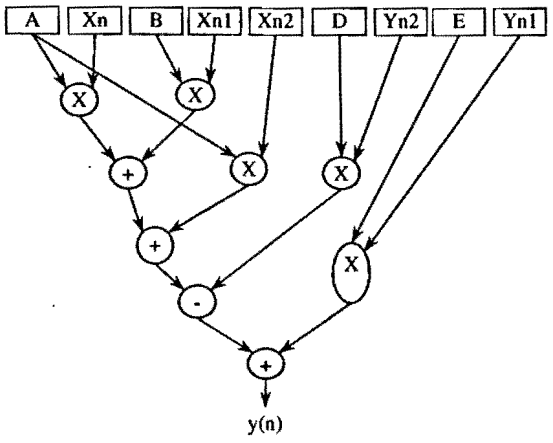

Figure 4. 6 Data Flow Graph

After the ASAP algorithm is applied, the timing diagram for the obtained schedule for the data flow graph under the resources constraint (which is configuration of the best variant) is shown in figure 4.7.

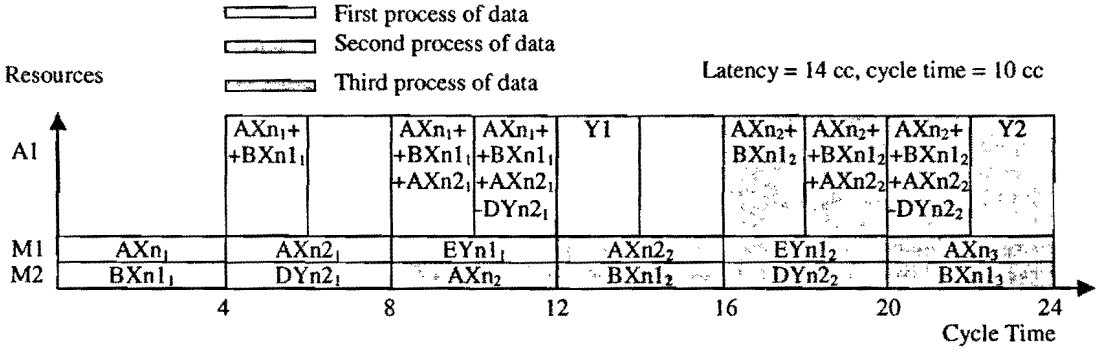

Figure 4. 7 Timing diagram for the schedule obtained for best variant

The timing diagram clearly shows in each time slot, which operation should be done with which resource and in which time slot. For example, figure 4.7 shows that from the time 0 to 4 (c.c), the resource M1 should perform the multiply operation AX_{n1}.

However, the timing diagram does not show the data flow among different resources. In order to do so, a sequencing graph with binding information should be created (see figure 4.8) since the flow of data elements through different operators in the data path can be visualized with the help of sequencing graphs [25].

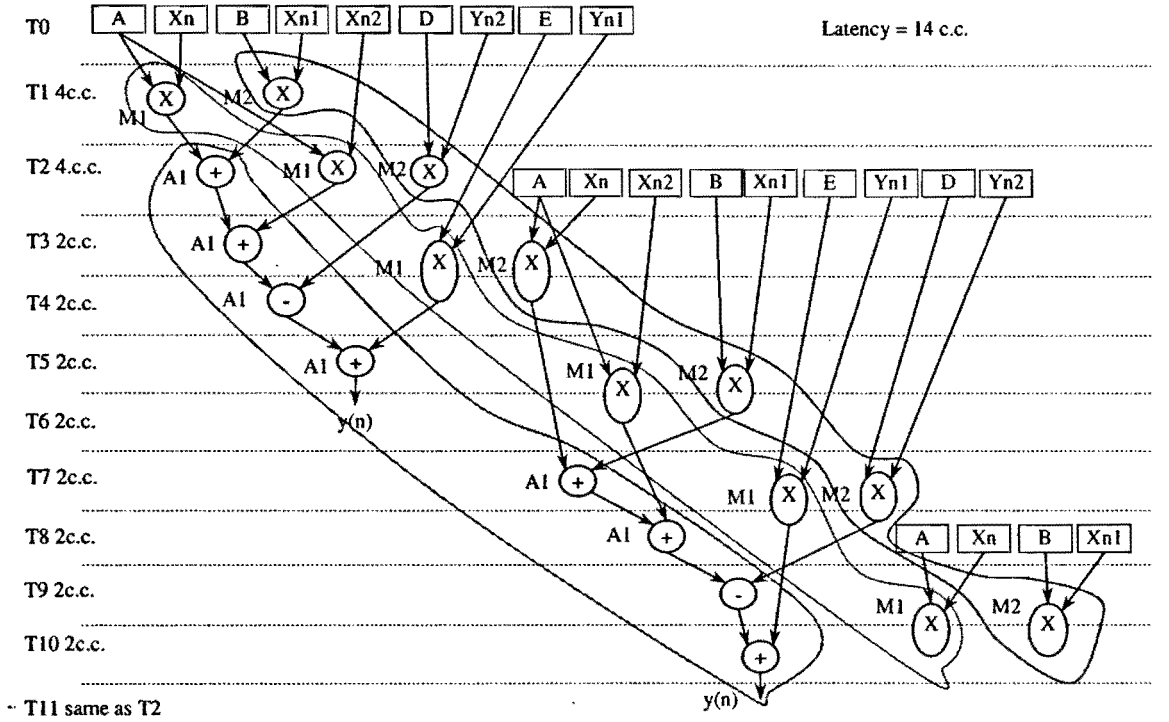

Figure 4. 8 Sequencing graph with binding information

Sequencing graph figure 4.8 clearly shows the data flow through different resources at different time slot. With that information, the connections among different hardware resources can be easily deduced. First, registers must be added into the sequencing graph in order to keep the temporary data before it is used by the next resources.

The function of registers is to perform storage of data and the functions of the wires are to interconnect the different discrete components [25]. The register 1 has been added in time slot T3 because the results of the multiplier (M2) at time slot T2 is not being used until time slot T4. Similarly for the implementation of the pipelining, the register 2 has been added because the output of the multiplier (M2) at time slot T4 is not used until time slot T7.

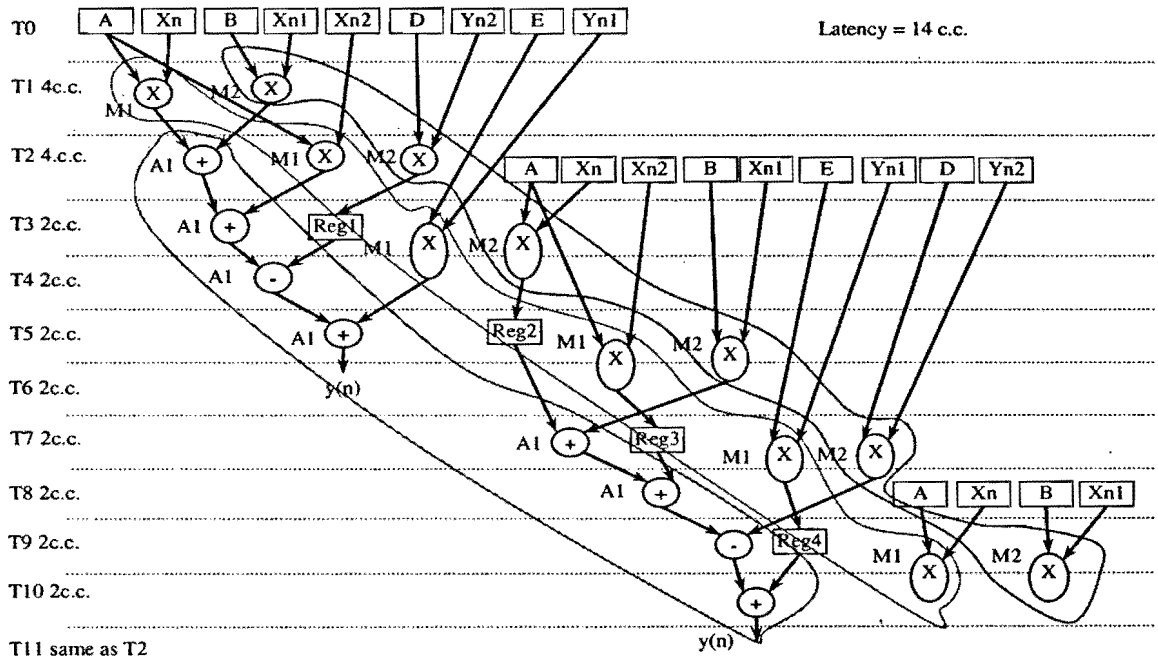

Figure 4. 9 Sequencing graph with data registers showing the portion of pipelining

Furthermore, registers 3 and 4 have been added in the pipelining portion because the output of multiplier (M1) and multiplier (M2) are not being read until the time slot T8 and T10 respectively. The sequencing graph with data registers is shown in figure.4.9.

4.11. Determination of the Multiplexing Scheme

The binding of the resources shown in Figure.4.9 enable formalization of a methodology of incorporating the multiplexers and demultiplexers into the data path circuit of the system. Multiplexing scheme is one of the most important stages for developing the structure of data path in high level synthesis design flow. This procedure represents each resource of a system with respective inputs, outputs, operations and the necessary interconnections and highlights the actual usage of resources at different times. The scheme table acts as an important guide for the designer to develop a systems block diagram before developing the control unit structure for the data path. This scheme prevents the designer from making any errors in the final hardware structure.

In this section, three resources perform different functions for the circuit, namely one adder/subtractor and two multipliers. A multiplexing scheme table was developed for each as shown in table 4.4 , 4.5, and 4. 6 respectively.

Table 4. 4 Multiplexing table for resource adder/sub (A1)

Time Slot	Opn	Input 1	Input 2	Output
0	-------	-------	-------	-------
1		M1out	M2out	
2	(+)	A1out	M1out	A1in
3	(+)	A1out	Reg1	A1in
4	(-)	A1out	M1out	A1in
5	(+)	-------	--------	RegY
6	-------	Reg2	M2out	-------
7	(+)	A1out	Reg3	A1in
8	(+)	A1out	M2out	A1in
9	(-)	A1out	Reg4	A1in
10	(+)	M1out	M2out	RegY

Table 4. 5 Multiplexing scheme for multiplier resource (M1)

Time Slot	Opn	Input 1	Input 2	Output
0	-------	RegA	RegXn	-------
1	(*)	RegA	RegXn2	A1in
2	(*)	RegE	RegYn1	A1in
3 and 4	(*)	RegA	RegXn2	A1in
5 and 6	(*)	RegE	RegYn1	Reg3
7 and 8	(*)	RegA	RegXn	Reg 4
9 and 10	(*)	RegA	RegXn2	A1in

Table 4. 6 Multiplexing scheme for multiplier resource (M2)

Time slot	Opn	Input 1	Input 2	Output
0	-------	Reg B	Reg Xn1	------
1	(*)	Reg D	Reg Yn2	A1in
2	(*)	Reg A	Reg Xn	Reg 1
3 and 4	(*)	Reg B	Reg Xn1	Reg 2
5 and 6	(*)	Reg D	Reg Yn2	A1in
7 and 8	(*)	Reg B	Reg Xn1	A1in
9 and 10	(*)	Reg D	Reg Yn2	A1in

Once the multiplexing scheme is created, multiplexers and demultiplexers can be easily constructed and assigned to their respective inputs and outputs.

4.12. Development of the System Block Diagram

After the multiplexing scheme has been successfully developed, the next stage of the design flow is to develop the system block diagram. The system block diagram consists of two divisions, data path and the control path. The data path is responsible for the flow of data through the data buses to the different components in the circuit block diagram. The data path consists of functional unit for execution of operations, registers for storage of data, multiplexers for preparation of input data, demultiplexers for allocation of output data, and memory elements

such as latches for the sinking of data for next stage. In block diagram for the sample application there are three resources (two adder/subtractors, and one multiplier) to execute their assigned operation that they are in charge of. Another division of the system block diagram is the control unit or the controller. The control unit is basically a centralized control unit that controls the entire data path and provides the control signals for the necessary timing and synchronization purpose which is required by the data traveling through the different components.

Based on the multiplexing scheme the block diagram of the data path circuit was constructed in Figure. 4.10 for the sample application.

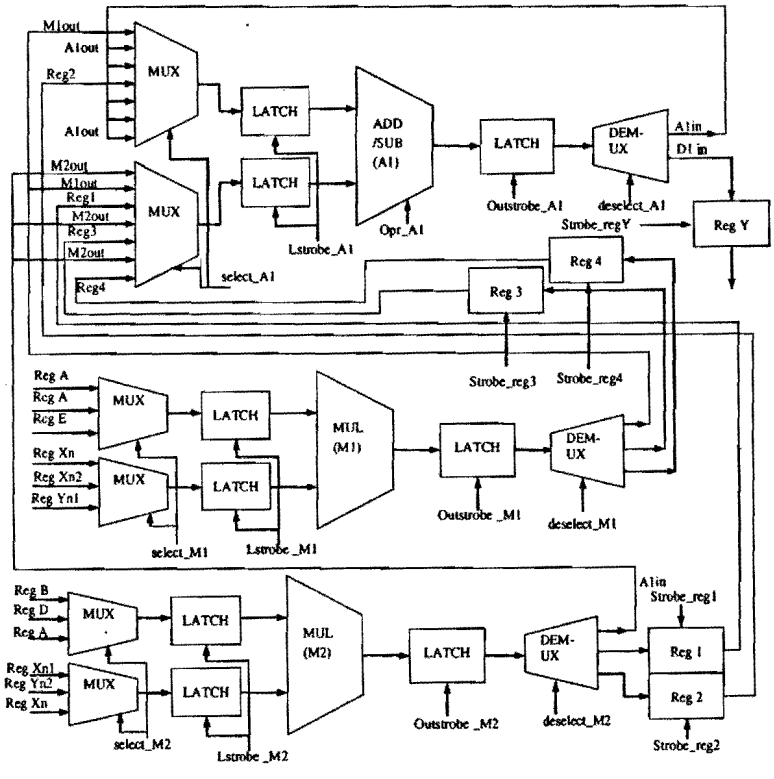

Figure 4. 10 Block diagram of the Data path circuit

4.13. Development of the Centralized Control Unit

The control unit plays an important role in allowing the system to perform the objective function. The control unit acts as a finite state machine that changes its state or the value of output control signals according to the requirement of the various elements of the data path at different instant of time, and it is responsible for the coordination of different elements in the data path of the system. Moreover, the control unit is also the circuit component, which is responsible for any mis-co-ordination in timing among the various elements of the data path.

The main function of the controller is to activate or deactivate the different elements of the data path based on the timing specification determined for the objective function. For synchronous functioning of all the data elements in the system, the controller has to respond to the requirement exactly at the right instance. Failure to activate or deactivate any block of element in the data path will result in fatal consequences in the system output.

In this thesis, Very high speed Hardware Description Language (VHDL) [22] has been used for the design of the control unit. The timing specification table for the data path circuit is shown in Table 4.7, in which clock cycle values are placed on the Y-axis and the control signals are placed on the X-axis. At every clock cycle (or the value of the counter), the values of different control signals can be clearly observed, and the control structure can be easily described in any hardware description language.

Table 4. 7 Timing specification of data path

	A1						M1				M2				Register strobes					
Time slot	CNT	Sel	De-sel	Add/sub	Latch	Outstrb	Sel	De-sel	Latch	Outstrb	Sel	De-sel	Latch	Outstrb	Reg1	Reg 2	Reg 3	Reg4	Latch data	Reg Y
T0	0	000	0	0	0	0	00	00	0	0	00	00	0	0	0	0	0	0	0	0
	1						00				00								1	
T1	2							00	1	0		00	1	0					0	
	3	000					01		0		01		0							
	4																			
	5									1				1						
T2	6		0	1	1	0		00	1	0		01	1	0						
	7	001			0		10		0		10		0							
	8																			
	9					1				1				1						
T3	10		0	1	1	0		00	1	0		10	1	0	1					
	11	010			0	1	01		0		00		0		0					
T4	12		0	0	1	0														
	13	001			0	1				1				1						
T5	14		1	1	1	0		01	1	0		00	1	0		1				
	15				0	1	10		0		01		0			0				
T6	16	011																		1
	17									1				1						0
T7	18		0	1	1	0		10	1	0		00	1	0			1			
	19	100			0	1	00		0		00		0				0			
T8	20		0	1	1	0													1	
	21	101			0	1				1				1					0	
T9	22		0	0	1	0		00	1	0		00	1	0				1		
	23	110			0	1	01		0		01		0					0		
T10	24		1	1	1	0														
	25	000			0	1				1				1						
T11	26		0	1	1	0		00	1	0		01	1	0						1
	7	001			0		10		0		10		0							0

4.14. Schematic Structure Development for the Whole System and Verification

After successful completion of all the above steps, the schematic structure of the device is ready for development in any of the available synthesis tools. The schematic structure comprises

the data path unit and the control unit. First, all the components in the data path are described and implemented in VHDL before verification. Then the schematic structure of the whole device is designed and implemented in Xilinx Integrated Software Environment (ISE) version 9.2i [27] [28]. The schematic structure of the whole device as designed in Xilinx ISE 9.2i is shown in Figure. 4.11.

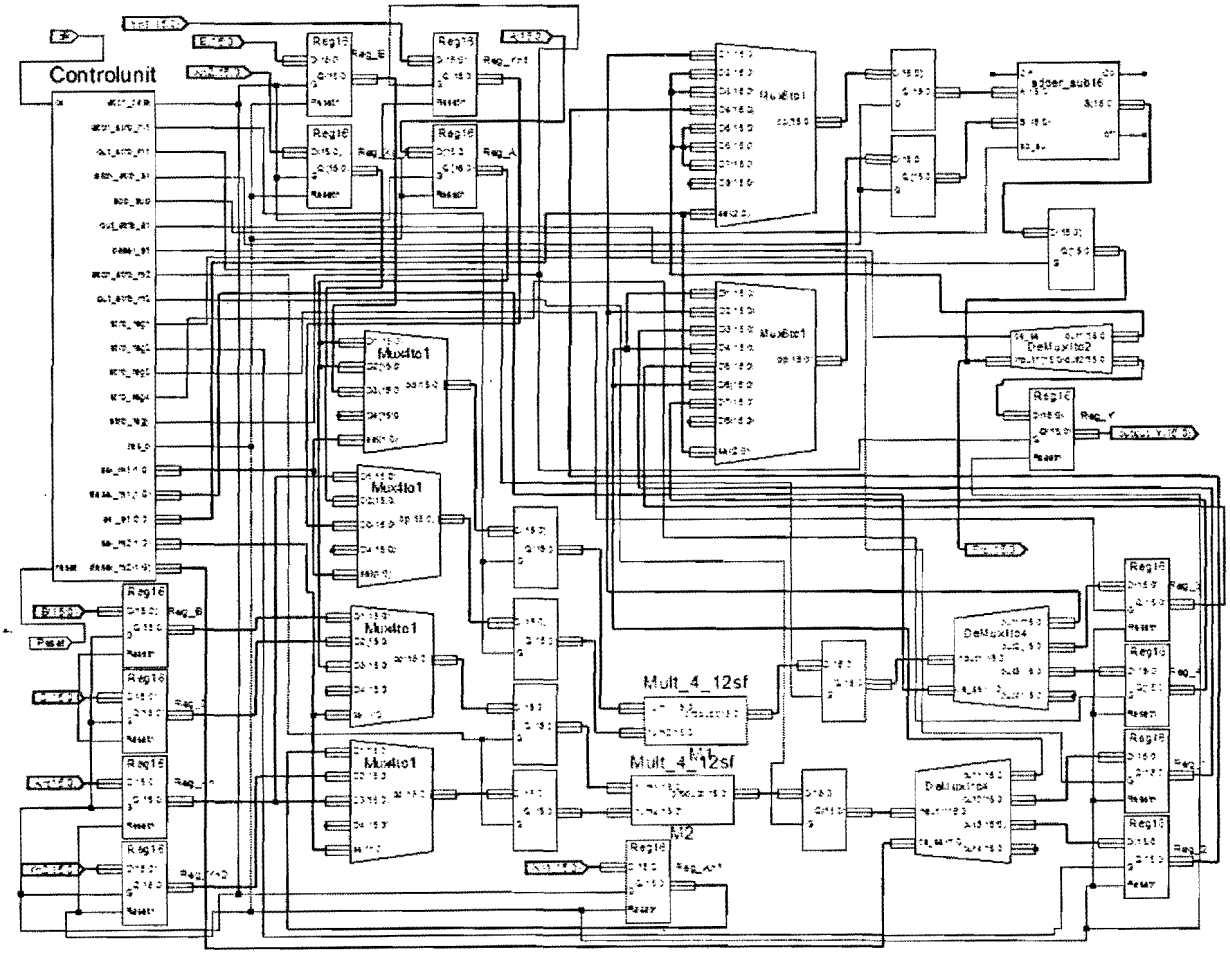

Figure 4. 11 Schematic view of the system (Xilinx ISE 9.2i)

Next, the verification of the designed system is carried out in Xilinx ISE simulator. The designed system is checked for wide array of input vectors and the simulation results indicate that the design is successfully accomplished. The successful result of the simulation suggests that the designed system gives the expected output. The design implementation in the Spartan 3E FPGA [34] suggests that the device is working perfectly and is in compliance with all the specifications provided. After the simulation, the design is imported in Synopsys tool for flattening. After flattening, the design is carried out in Cadence Encounter SOC and IC custom design tool respectively for floor planning, power planning, placement, routing and layout.

Chapter 5

Experimental Results and Analysis

The proposed DSE approach was tested on several different size benchmarks including many large size benchmarks from the NCSU CBL high-level synthesis benchmark suite [29], such as *Elliptic,* which is an elliptic wave filter, and *Diffeq,* which is a differential equation solver. Furthermore, DSP benchmarks such as discrete wavelet transformation (DWT) [30], autoregressive filter (ARF) [31] [32], and MPEG motion vectors (MMV) [33] were also selected for experiments.

For the different benchmarks, the proposed DSE approach which combines the priority factor and the fuzzy search technique is compared to the DSE approach which combines the priority factor and the binary search technique. The experiment results are shown in Table 5.1. The same specifications and constraints for all benchmarks were used during the comparison in order to

demonstrate the advantage of using the fuzzy search technique over the binary search which is used in other approaches, i.e. [3] during the DSE.

Table 5. 1 Comparison of Fuzzy search technique with binary search technique when they are applied to the hierarchy configuration tree during DSE

Bench marks	Number of architectural variants in the Design space	Total possible architectural variants in the design space for two parameters	Proposed hybridized approach with Priority Factor method and Fuzzy Search (Number of comparisons between variants)			Priority Factor method with Binary Search (Number of comparisons between variants)			Speed up compared with Binary Search
			Area	Texe	Total variants	Area	Texe	Total variants	
IIR Digital Chebyshev Filter	27	54	2	5	7	4	5	9	22.22%
IIR Digital Filter 1	32	64	2	4	6	5	5	10	40.00%
IIR Digital Filter 2	36	72	2	6	8	6	6	12	33.33%
IIR Digital Filter 3	40	80	2	4	6	5	6	11	45.45%
Elliptic Wave Filter (EWF) [29]	78	156	2	5	7	6	6	12	41%
Differential Equation Solver (HAL) [29]	90	180	4	9	13	6	7	13	--
Auto Regressive Filter (ARF) [31][32]	144	288	3	8	11	7	7	14	21%
Discrete Wavelet Transformation (DWT) [30]	216	432	4	8	12	8	8	16	25%

Figure 5.1 shows the comparison between the proposed fuzzy search technique and the binary search technique for the selected benchmarks. The experimental result indicates that the proposed fuzzy search technique can search the optimum variant faster than the binary search

technique when they are applied to the hierarchy configuration tree in the DSE. In other words, the proposed hybrid approach is capable of achieving higher speed up compared to the exhaustive search. For example, for the benchmark IIR digital filter 3, the speed up of up to 45.45% and 92.50% are achieved compared to the binary search and the exhaustive analysis respectively.

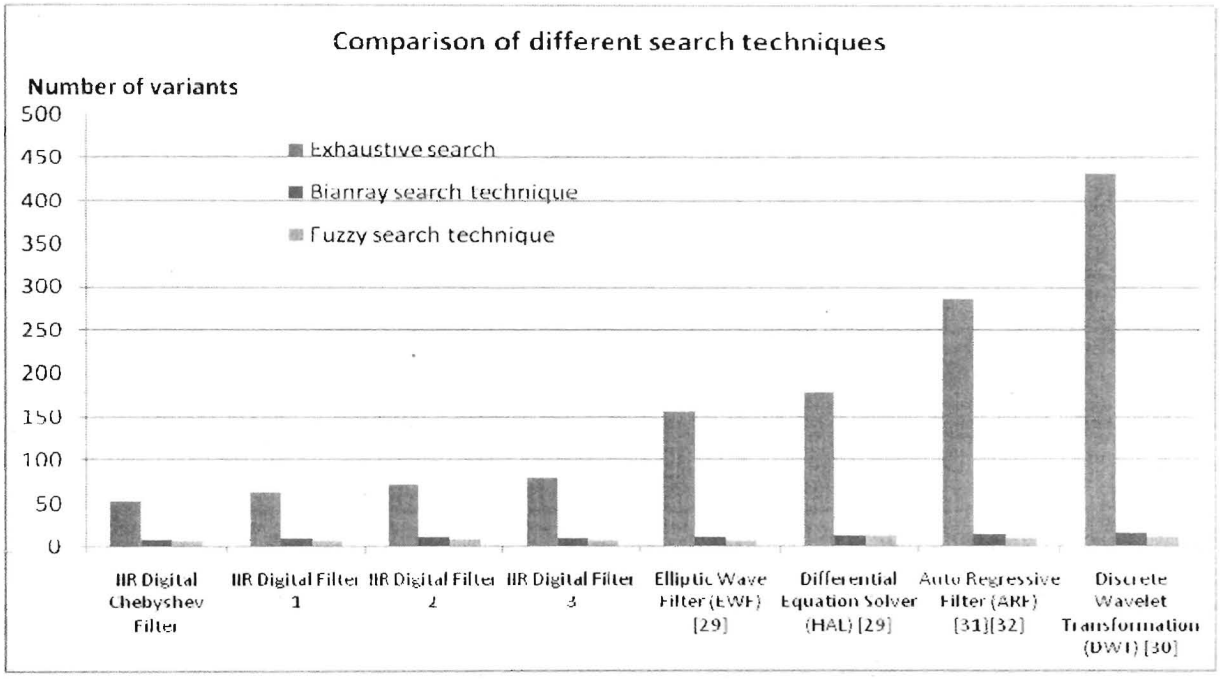

Figure 5. 1 Comparsion of different search techniques

The proposed approach is a combination of the priority factor method and the fuzzy search technique. This combination will improve the speed during the DSE process. The experiment results compared with the current approach [3] is shown in Table 5.2, and the Figure 5.2 shows the number of variants analyzed during the DSE process when different approaches were used

for the selected benchmarks. The comparison and speed up results in Table 5.2 indicate that adding the fuzzy search technique to the priority factor method for DSE increases the speed of the exploration process compared to the current approach [3] which is also based on the Pareto optimal analysis.

For example, the proposed hybrid approach yields speedup ranging from 26% to 47% for the selected benchmarks when compared to [3]. Acceleration of over 96% was also obtained compared to the exhaustive search for Discrete Wavelet Transformation (DWT) benchmark as seen in table 5.2.

Table 5. 2 Experimental results of the proposed hybridized DSE approach compared with the current approach

Benchmark applications	Total possible architecture in the design space for exhausted search	Existing approach with binary search [3] (Number of variants analyzed)	Proposed hybrid approach (Number of variants analyzed)	Percentage speed up compared to existing approach [3]	Percentage speed up compared to exhaustive approach
IIR Digital Chebyshev Filter	54	18	11	38.89%	79.63%
IIR Digital Filter 1	64	17	10	41.18%	84.38%
IIR Digital Filter 2	72	22	12	45.45%	83.33%
IIR Digital Filter 3	80	19	10	47.37%	87.50%
Elliptic Wave Filter (EWF) [29]	156	19	11	42.10%	92.95%
Differential Equation Solver (HAL) [29]	180	23	17	26.08%	90.56%
Auto Regressive Filter (ARF) [31][32]	288	24	15	37.5%	94.79%
Discrete Wavelet Transformation (DWT) [30]	432	26	16	38.46%	96.30%

It can be seen that the proposed priority factor technique combined with fuzzy search provides increased acceleration in the design space exploration during the high level synthesis. The above analysis reveals that the proposed approach is able to provide massive acceleration in the design space exploration while simultaneously maintaining the accuracy needed in architecture selection.

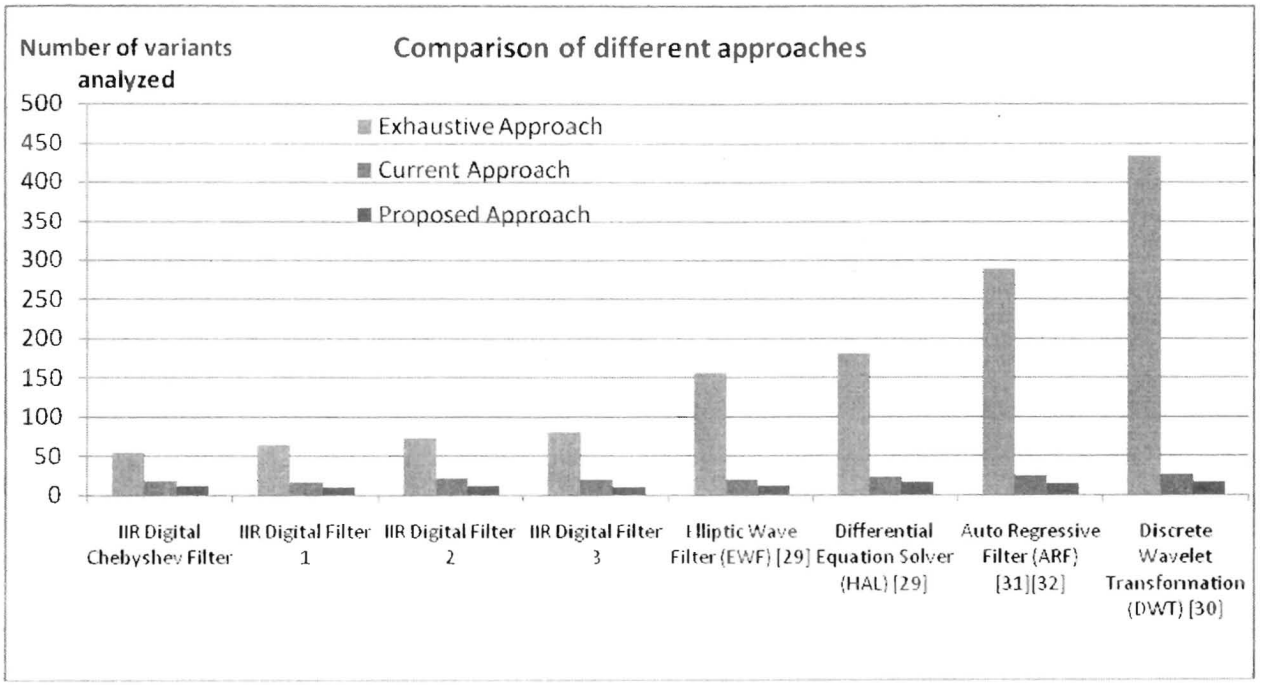

Figure 5. 2 Comparison of different approaches

According to the design flow, an optimized filter satisfying all the constraints was successfully designed. In Figure 5.3, the simulation result of the design filter shows that when different values of input vectors were applied to the input terminal, the designed filter can always provide the correct corresponded output value as expected. With the help of IC design tools, the final IC layout is shown in Figure 5.4.

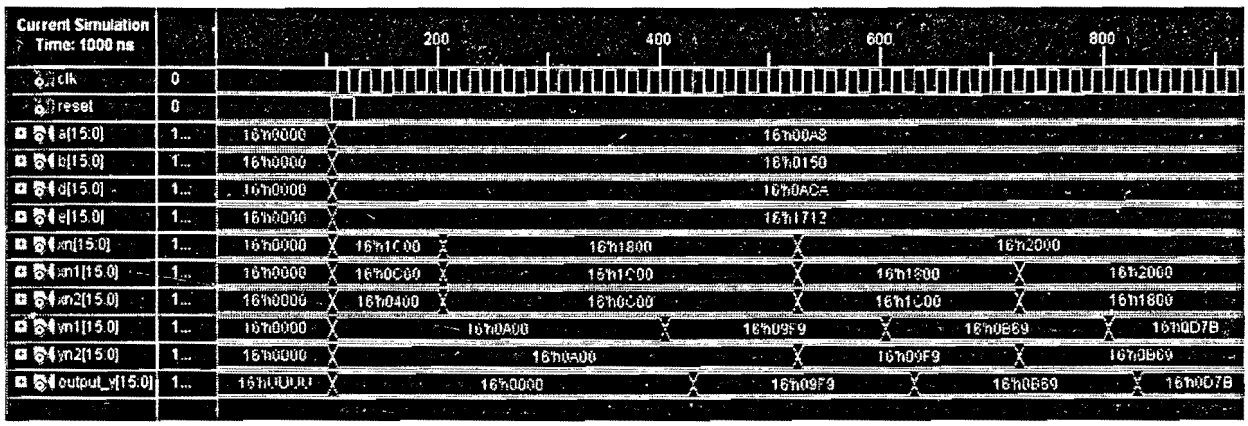

Figure 5. 3 Simulation results for the implemented filter

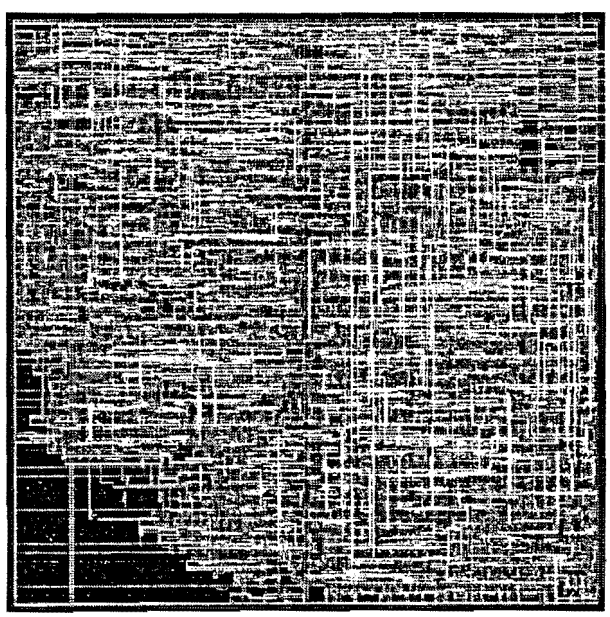

Figure 5. 4 Final routing of the chip (Cadence encounter SoC)

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This thesis presents a novel framework for the design space exploration using the priority factor method hybridized with a new fuzzy search mechanism. Furthermore, a new high level synthesis design flow with multi parametric optimization objective using the proposed DSE approach is also introduced. The proposed hybrid approach offers faster DSE approach as compared to the other approaches for DSE in high level synthesis. This approach first organizes the architecture design space in ascending or descending order and then uses the fuzzy search technique to assess and select the most suitable architecture according to the user specifications. The concept of fuzzy membership value is used for developing the fuzzy search technique. Furthermore, obtaining the optimal architecture with the minimum exploration time and high precision is highly desirable for efficient DSE. The proposed DSE method shows the improvement in speedup compared to the current existing approach. Moreover, the whole

process of DSE along with the high level synthesis design flow allows easy automation of the methodology useful for the current high level synthesis tools.

6.2 Future work

In order to get an efficient design according to the design flow, a number of algorithms need to be applied in the design steps within the proposed design flow. Optimization on each sub stage of the design flow can definitely optimize the whole design. Highly efficient algorithms for the sub stages such as scheduling, and binding etc. can be used along with the proposed DES approach and the design flow. With all the necessary algorithms provided, the proposed design flow can also be easily automated.

Publications

1. **Zhipeng Zeng**, Reza Sedaghat, Anirban Sengupta, "A multi parametric optimized High Level Synthesis design flow with a Hybrid approach to rapid Design Space Exploration using Fuzzy Search technique", Journal of Systems architecture, Elsevier, First Revision, 2009,Submission no: JSA-D-09-00092
2. **Zhipeng Zeng**, Reza Sedaghat, Anirban Sengupta, "A multi parametric optimized High level synthesis Design Flow for multi objective VLSI and SoC designs", Integration VLSI journal, Elsevier, 2009, Submission no: VLSI-D-09-00120 (submitted) Refereed Conferences
3. Anirban Sengupta, Reza Sedaghat, **Zhipeng Zeng**, "A High Level Synthesis design flow with a novel approach for Efficient Design Space Exploration in case of multi parametric optimization objective", Microelectronics Reliability, Elsevier, In press, Corrected Proof, 2009, Submission no: MR-D-09-00308.
4. Anirban Sengupta, Reza Sedaghat, **Zhipeng Zeng**, "A High Level Synthesis Design Flow from ESL to RTL with multi-parametric optimization objective", IETE Journal of Research, 2009, Submission no: IETE JR_523_09 (submitted)
5. **Zhipeng Zeng**, Reza Sedaghat, Anirban Sengupta, "A Novel Framework of Optimizing Modular Computing Architecture for multi objective VLSI designs", IEEE 21st International Conference on Microelectronics (ICM), 2009, Accepted for publication in the Conference Proceedings, In press, Article # 1569230822, To be presented on December 21, 2009.

6. **Zhipeng Zeng**, Reza Sedaghat, Anirban Sengupta, "A Framework for Fast Design Space Exploration using Fuzzy search for VLSI Computing Architectures", IEEE International Symposium on Circuits and Systems (ISCAS), 2009, submitted.
7. Anirban Sengupta, Reza Sedaghat, **Zhipeng Zeng**, "Hardware Efficient Design of speed optimized Power stringent Application Specific Processor", IEEE 21st International Conference on Microelectronics (ICM), 2009, Accepted for publication in the Conference Proceedings, In press, Article # 1569230826, To be presented on December 22, 2009.
8. Summit Sehgal, Reza Sedaghat, Anirban Sengupta, **Zhipeng Zeng**, "Multi Parametric Optimized Architectural Synthesis of an Application Specific Processor", IEEE 14th International CSI Computer Conference, 2009, Accepted for publication in the conference proceedings (in press), Article id:13.
9. Anirban Sengupta, Reza Sedaghat, **Zhipeng Zeng**, "Rapid Design Space Exploration for multi parametric optimization of VLSI designs", IEEE International Symposium on Circuits and Systems (ISCAS), 2009, submitted.

References

[1] P.Coussy, A. Morawiec, (Eds.) "High-Level Synthesis From Algorithm to Digital Circuit" ISBN 978-1-4020-8587-1 (Print) 978-1-4020-8588-8 (Online) page 15

[2] S. Mohanty, V. K. Prasanna, S. Neema, and J. Davis, "Rapid Design Space Exploration Of Heterogeneous Embedded Systems Using Symbolic Search And Multi-Granular Simulation". In Proceedings Of The Joint Conference On Languages, Compilers And Tools For Embedded Systems: Software And Compilers For Embedded Systems. 2002. pp.18–27

[3] L. Kirischian, V. Geurkov, V. Kirischian, and I. Terterian, 'Multi-parametric optimisation of the modular computer architecture', Int. J.Technology, Policy and Management, 2006, Vol. 6, No. 3, pp.327–346.

[4] I. Das, "A preference ordering among various Pareto optimal alternatives". Structural and Multidisciplinary Optimization, 18(1):30–35, Aug. 1999.

[5] V. Krishnan, and S. Katkoori, "A Genetic Algorithm for the Design Space Exploration of Datapaths During High-Level Synthesis", IEEE Transactions on Evolutionary Computation, vol. 10, no. 3, June 2006.

[6] A. Giuseppe, C. Vincenzo, G. Alessandro, D. Nuovo, M. Palesi, and D. Patti, "Efficient design space exploration for application specific systems-on-a-chip", Journal of Systems Architecture 53 (2007) Pages: 733–750.

[7] E. Torbey and J. Knight, "High-level synthesis of digital circuits using genetic algorithms," in Proc. Int. Conf. Evol. Comput., May 1998, pp.224–229.

[8] E. Torbey and J. Knight, "Performing scheduling and storage optimization simultaneously using genetic algorithms," in Proc. IEEE Midwest Symp. Circuits Systems, 1998, pp. 284–287.

[9] J. C. Gallagher, S. Vigraham, and G. Kramer,"A family of compact genetic algorithms for intrinsic evolvable hardware," IEEE Trans. Evol. Comput., vol. 8, no. 2, pp. 1–126, Apr. 2004.

[10] Zitzler, E., Laumanns, M., and L. Thiele, 2002. Spea2: "Improving The Strength Pareto Evolutionary Algorithm For Multiobjective Optimization". In Proceedings Of The Conference On Evolutionarymethods For Design, Optimisation, And Control. 19–26.

[11] A. D. Pimentel, C. Erbas, and S. Polstra,. "A Systematic Approach To Exploring Embedded System Architectures At Multiple Abstraction Levels". Ieee Trans. Comput. 2006, vol 55, no. 2, pp.99–112

[12] M. Kim, S. Banerjee, N. Dutt, and N. Venkatasubramanian. "Design space exploration of real-time multi-media MPSoCs with heterogeneous scheduling policies", In Proceedings of the International Conference on Hardware/Software Codesign (CODES+ISSS). 2006, pp.16–21.

[13] S. Mamagkakis, D. Atienza, C. Poucet, F. Catthoor, D. Soudris, and J. M. Mendias. "Automated Exploration Of Pareto-Optimal Configurations In parameterized dynamic

memory allocation for embedded systems", In Proceedings of the Design, Automation and Test in Europe Conference (DATE). pp. 874–875, 2006.

[14] T. Gupta, P. Sharma, M. Balakrishnan, S. Andmalik, "Processor evaluation in an embedded systems design environment", In Proceedings of the 13th International Conference on VLSI Design. pp.98–103. 2000

[15] M. Lukasiewycz, M. Glaß, C. Haubelt, J. Teich, "Efficient symbolic multi-objective design space exploration", Asia and South Pacific Design Automation Conference, Proceedings of the 2008 Asia and South Pacific Design Automation Conference, Pages 691-696,2008

[16] A. D. Pimentel, "The Artemis workbench for system-level performance evaluation of embedded systems", International Journal of Embedded Systems,Vol 3, Num 3, Pages:181 - 196, 2008

[17] K. Sigdel, M. Thompson, A. D. Pimentel, and T. Stefanov, "System-Level Design Space Exploration of Dynamic Reconfigurable Architectures", Springer, Vol 5114, Pages: 279-288, 2008

[18] C. Lee, S. Kim, and S. Ha, "A Systematic Design Space Exploration of MPSoC Based on Synchronous Data Flow Specification", Journal of Signal Processing Systems, Springer, 2009

[19] C.T. Hwang, J.H. Lee, and Y.C. Hsu, "A Formal Approach to the Scheduling Problem in High Level Synthesis", IEEE Transactions on Computer-Aided Design, Vol. 10, no. 4, April 1991.

[20] R. Larson, R. P. Hostetler, B. H. Edwards, "Calculas with Analytic Geometry", Houghton Mifflin Company, Eighth Edition, 2006, Pages: 918-919

[21] L. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning", information sciences, 1975, American Elsevier Publishing Company.

[22] S. Salivahanan, A. Vallavaraj and C. Gnanapriya, "Digital Signal Processing", Tata McGraw-Hill Publishing Company Limited, pp. 439- 444. 2006.

[23] P.G. Paulin and J. P. Knight, "Scheduling and Binding Algorithms for High-Level Synthesis, 26th conference on Design Automation, 1988, Pages: 1-6.

[24] S.P. Mohanty, N. Ranganathan, E. Kougianos and P. Patra, "Low-Power High-Level Synthesis for Nanoscale CMOS Circuits", Chapter- High-Level Synthesis Fundamentals, Springer US, 2008

[25] G. D. Micheli, "Synthesis and Optimization of Digital Systems", McGraw-Hill Inc., pp580, 1994

[26] I. Ahmad, M.K. Dhodhi, C.Y.R. Chen, "Integrated scheduling, allocation and module selection for design-space exploration in high-level synthesis", IEE Proceedings, Computer and Digital Techniques, Vol. 142, Issue. I, January 1995, Pages: 65-71

[27] ISE 9.2i Quick Start Tutorial, Xilinx ISE 9.2i , Software Manuals and Help, http://www.xilinx.com/support/sw_manuals/xilinx92/download/

[28] www.xilinx.com/itp/xilinx92/books/manuals.pdf

[29] http://www.cbl.ncsu.edu/benchmarks/.

[30] R. Jain, P.R. Panda, "An efficient pipelined VLSI architecture for lifting-based 2d-discrete wavelet transform". In Proceedings of the International Symposium on Circuits and Systems (ISCAS), pp. 1377– 1380 (2007)

[31] A. Antola, F. Ferrandi, V. Piuri, and M. Sami, "Semiconcurrent error detection in data paths", IEEE Transactions on Computers vol 50, (5), 2001. pp. 449–465.

[32] A. Antola,, V. Piuri, M. Sami, "A low-redundancy approach to semi-concurrent error detection in datapaths," In Proceedings of the Design Automation and Test in Europe, 1998. pp. 266– 272

[33] Express: High-Level Synthesis Benchmarks. http://express.ece.ucsb.edu/benchmark/

[34] http://www.xilinx.com/publications/xcellonline/xcell_54/xc_ssinterface54.htm